Red Hat Certificate System 10

管理ガイド

Red Hat Certificate System 10.4 向けに更新

Last Updated: 2023-08-30

Red Hat Certificate System 10 管理ガイド

Red Hat Certificate System 10.4 向けに更新

Florian Delehaye
Red Hat Customer Content Services
fdelehay@redhat.com

Marc Muehlfeld
Red Hat Customer Content Services

Petr Bokoč
Red Hat Customer Content Services

Filip Hanzelka
Red Hat Customer Content Services

Tomáš Čapek
Red Hat Customer Content Services

Ella Deon Ballard
Red Hat Customer Content Services

法律上の通知

概要

本書では、Certificate System サブシステムのインストール、設定、および管理のあらゆる側面を説明します。また、ユーザーの追加、証明書の要求、更新、および失効、CRL の公開、スマートカードの管理などの管理タスクも説明します。本ガイドは、Certificate System の管理者を対象としています。

目次

第1章 RED HAT CERTIFICATE SYSTEM サブシステムの概要

すべての一般的な PKI 操作 (証明書の発行、更新、取り消しなど、キーのアーカイブと回復、CRL の公開と証明書ステータスの検証) は、Red Hat Certificate System 内の相互運用サブシステムによって実行されます。この章では、個々のサブシステムの機能と、それらが連携して堅牢でローカルな PKI を確立する方法を説明します。

1.1. 証明書に使用

証明書の目的は、信頼を確立することです。その使用法は、それが保証するために使用される信頼の種類によって異なります。提示者のアイデンティティーを確認するために、いくつかの種類の証明書が使用されたり、オブジェクトまたはアイテムが改ざんされていないことを確認したりするために使用されます。

証明書の使用方法、証明書の種類、または証明書の ID と関係の確立方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『証明書および認証』セクションを参照してください。

1.2. CERTIFICATE SYSTEM サブシステムのレビュー

Red Hat Certificate System は 5 つの異なるサブシステムを提供します。それぞれは、PKI デプロイメントのさまざまな側面に重点を置いています。これらのサブシステムは連携して、公開鍵インフラストラクチャー (PKI) を作成します。インストールされているサブシステムに応じて、PKI はトークン管理システム (TMS) または非トークン管理システムとして機能できます。サブシステム、TMS、および TMS 以外の環境の説明は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『Certificate System サブシステムのレビュー』セクションを参照してください。

Enterprise Security Client
Enterprise Security Client は、証明書、鍵、またはトークンで操作を実行しないため、サブシステムではありません。Enterprise Security Client は、ユーザーがスマートカードで証明書を簡単に管理できるようにするユーザーインターフェイスです。Enterprise Security Client は、証明書要求などのすべてのトークン操作をトークン処理システム (TPS) に送信し、認証局 (CA) に送信します。詳細については、『Red Hat Certificate System エンタープライズセキュリティークライアントでスマートカードの管理』を参照してください。

1.3. 証明書管理の概観 (非 TMS)

従来の PKI 環境は、ソフトウェアデータベースに保存されている証明書を管理する基本的なフレームワークを提供します。これは、スマートカードで証明書を管理しないため、**TMS 以外** の環境です。少なくとも、TMS 以外の環境では CA のみが必要ですが、TMS 以外の環境では OCSP レスポンダー と KRA インスタンスも使用できます。

このトピックに関する詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の以下のセクションを参照してください。

- 証明書の管理
- 単一 Certificate Manager の使用
- 鍵のプランニング: キーアーカイブおよびリカバリー
- 証明書要求の処理の分散
- クライアント OCSP 要求の分散

1.4. TOKEN MANAGEMENT SYSTEM (TMS) の概観

証明書システムは、証明書の作成、管理、更新、取り消しを行い、鍵のアーカイブおよび復元も行います。スマートカードを使用する組織の場合は、Certificate System に、トークン管理システムがあります。これは、鍵と要求を生成し、スマートカードに使用される証明書を受け取るために、確立された関係を持つサブシステムのコレクションです。

このトピックに関する詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の以下のセクションを参照してください。

- スマートカードとの連携 (TMS)
- スマートカードの使用

1.5. RED HAT CERTIFICATE SYSTEM サービス

ユーザータイプ (管理者、エージェント、監査ユーザー、エンドユーザー) に応じて、証明書やサブシステムの管理にはさまざまなインターフェイスがあります。各インターフェイスを通じて実行されるさまざまな機能の概要は、『ユーザーインターフェイス』 のセクションを参照してください。

パート I. RED HAT CERTIFICATE SYSTEM ユーザーインターフェイス

第2章 ユーザーインターフェイス

ユーザーロール (管理者、エージェント、監査ユーザー、エンドユーザー) に応じて、証明書やサブシステムの管理にはさまざまなインターフェイスがあります。

2.1. ユーザーインターフェイスの概要

管理者は、以下のインターフェイスを使用して、完全な Certificate System インストールと安全に対話できます。

- PKI コマンドラインインターフェイスおよびその他のコマンドラインユーティリティー
- PKI コンソールのグラフィカルインターフェイス
- Certificate System Web インターフェイス

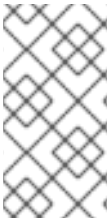
これらのインターフェイスには、TLS による Certificate System サーバーとの安全な通信に使用する前に設定が必要です。適切な設定なしでこれらのクライアントを使用することはできません。これらのツールの一部は、TLS クライアント認証を使用します。必要に応じて、必要な初期化手順にこれの設定が含まれます。使用するインターフェイスは、管理者の設定と機能によって異なります。これらのインターフェイスを使用する一般的なアクションは、本章の後半の残りのガイドに記載されています。

デフォルトでは、PKI コマンドラインインターフェイスは、ユーザーの ~/**.dogtag/nssdb**/ ディレクトリーにある NSS データベースを使用します。「pki CLI の初期化」では、NSS データベースを管理者の証明書およびキーで初期化する詳細な手順を説明します。PKI コマンドラインユーティリティーの使用例は、「pki CLI の使用」に記載されています。その他の例を以下に示します。

(他のユーザーロールの管理者として) 証明書システムとの干渉は、さまざまなコマンドラインユーティリティーを使用して、CMC 要求の送信、生成された証明書の管理などを行うことができます。これらについては、「AtoB」など、「コマンドラインインターフェイス」で簡単に説明します。これらのユーティリティーは、「**PKCS10Client** を使用した CSR の作成」 などの後のセクションで使用されています。

Certificate System Web インターフェイスを使用すると、Firefox Web ブラウザーから管理アクセスが可能になります。「ブラウザーの初期化」 は、クライアント認証の設定手順を説明します。「Web インターフェイス」 の他のセクションでは、証明書システムの Web インターフェイスの使用を説明します。

Certificate System の PKI コンソールはグラフィカルインターフェイスです。**この機能は非推奨になりました。**「**pkiconsole** の初期化」 では、このコンソールインターフェイスを初期化する方法を説明します。「CA、OCSP、KRA、および TKS サブシステムに対する **pkiconsole** の使用」 では、その使用の概要を説明します。「Java ベースの管理コンソールを使用した証明書の登録プロファイルの管理」などの後のセクションでは、特定の操作について詳しく説明します。



注記

PKI コンソールセッションを終了するには、**Exit** (終了) ボタンをクリックします。Web ブラウザーセッションを終了するには、ブラウザーを閉じます。コマンドラインユーティリティーは、アクションを実行してプロンプトに返すとすぐにそれ自身を終了するため、セッションを終了するには、管理者の部分でアクションは必要ありません。

2.2. クライアント NSS データベースの初期化

Red Hat Certificate System では、特定のインターフェイスが TLS クライアント証明書認証 (通常は認証) を使用してサーバーにアクセスしなければならない場合があります。サーバー側の管理タスクを実行する前に、以下を行う必要があります。

1. クライアント用の NSS データベースを準備します。これは、新規データベースか、または既存のデータベースにすることができます。
2. CA 証明書チェーンをインポートして信頼します。
3. 証明書と対応するキーがあります。NSS データベースで生成したり、PKCS #12 ファイルから他の場所からインポートしたりできます。

ユーティリティーに基づいて、NSS データベースを適宜初期化する必要があります。以下を参照してください。

- 「pki CLI の初期化」
- 「**pkiconsole** の初期化」
- 「ブラウザーの初期化」

2.3. グラフィカルインターフェイス

重要

pkiconsole が非推奨になりました。

Certificate System のコンソール **pkiconsole** は、Administrator ロール権限を持つユーザーがサブシステム自体を管理するためのグラフィカルインターフェイスです。これには、ユーザーの追加、ログの設定、プロファイルおよびプラグインの管理、および内部データベースなどの多くの機能が含まれます。このユーティリティーは、クライアント認証を使用して TLS 経由で Certificate System サーバーと通信し、リモートでサーバーを管理するために使用できます。

2.3.1. pkiconsole の初期化

pkiconsole インターフェイスを初めて使用するには、新しいパスワードを指定し、以下のコマンドを使用します。

```
$ pki -c password -d ~/.redhat-idm-console client-init
```

このコマンドは、~/**.redhat-idm-console**/ ディレクトリーに新しいクライアントの NSS データベースを作成します。

CA 証明書を PKI クライアント NSS データベースにインポートするには、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『NSS データベースへの証明書のインポート』を参照してください。

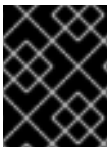
新しいクライアント証明書を要求するには、*5章証明書の要求、登録、および管理* を参照してください。

以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

```
$ openssl pkcs12 -in file -clcerts -nodes -nokeys -out file.crt
```

『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『証明書/キー暗号化トークンの管理』セクションに記載の手順に従って、管理クライアント証明書の検証およびインポートを行います。

```
$ PKICertImport -d ~/.redhat-idm-console -n "nickname" -t ",," -a -i file.crt -u C
```



重要

CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

既存のクライアント証明書とそのキーをクライアント NSS データベースにインポートするには、次を実行します。

```
$ pki -c password -d ~/.redhat-idm-console pkcs12-import --pkcs12-file file --pkcs12-password
pkcs12-password
```

以下のコマンドを使用して、クライアント証明書を確認します。

```
$ certutil -V -u C -n "nickname" -d ~/.redhat-idm-console
```

2.3.2. CA、OCSP、KRA、および TKS サブシステムに対する pkiconsole の使用

Java コンソールは、CA、OCSP、KRA、および TKS の 4 つのサブシステムで使用されます。コンソールには、ローカルにインストールされた **pkiconsole** ユーティリティーを使用してアクセスできます。コマンドにはホスト名、サブシステムの管理 TLS ポート、特定のサブシステムタイプが必要なため、あらゆるサブシステムにアクセスできます。

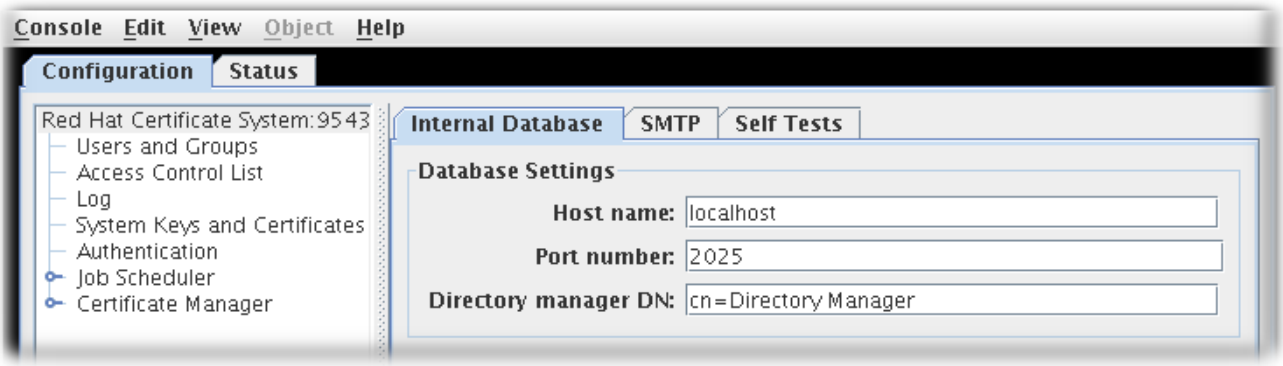
```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

DNS が設定されていない場合は、IPv4 アドレスまたは IPv6 アドレスを使用して、コンソールに接続できます。以下に例を示します。

```
https://192.0.2.1:8443/ca
https://[2001:DB8::1111]:8443/ca
```

これにより、図2.1「Certificate System コンソール」 にあるようにコンソールが開きます。

図2.1 Certificate System コンソール



Configuration タブは、名前が示すように、サブシステムのすべての設定を制御します。このセクショ

ンで利用可能な選択肢は、インスタンスがどのサブシステムタイプであるかによって異なります。CA にはジョブ、通知、および証明書登録認証の追加設定があるため、CA にはほとんどのオプションがあります。

すべてのサブシステムには 4 つの基本的なオプションがあります。

- ユーザーおよびグループ
- アクセス制御リスト
- ログ設定
- サブシステム証明書 (セキュリティードメインや監査署名など、サブシステムに発行した証明書)

Status タブには、サブシステムによってメンテナンスされるログが表示されます。

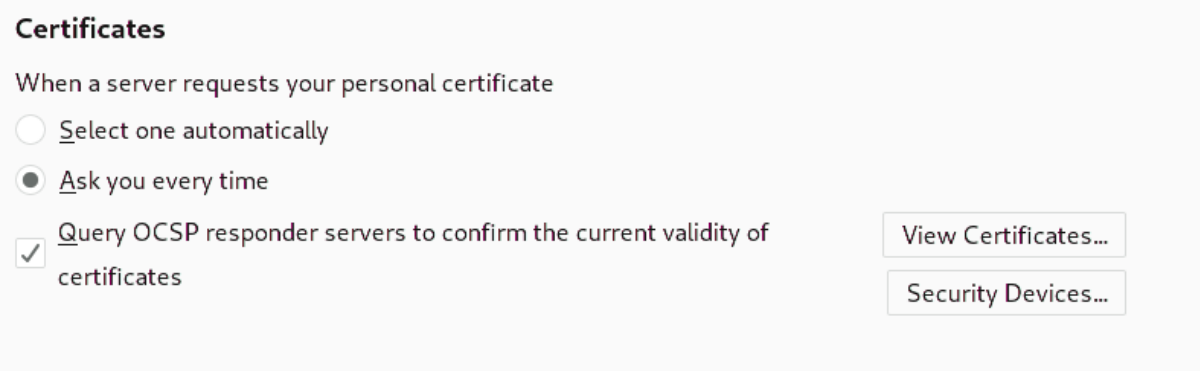
2.4. WEB インターフェイス

2.4.1. ブラウザーの初期化

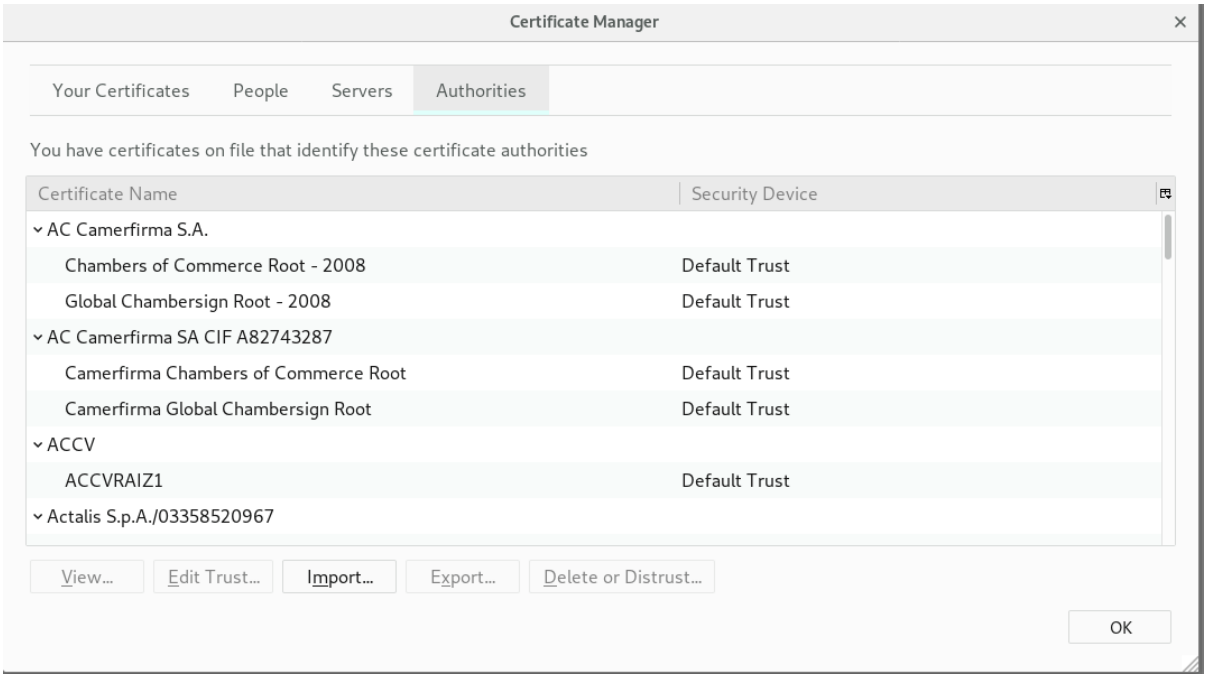
本セクションでは、Firefox が PKI サービスにアクセスするためのブラウザーの初期化を説明します。

CA 証明書のインポート

1. **Menu → Preferences → Privacy & Security→ View certificates**をクリックします。



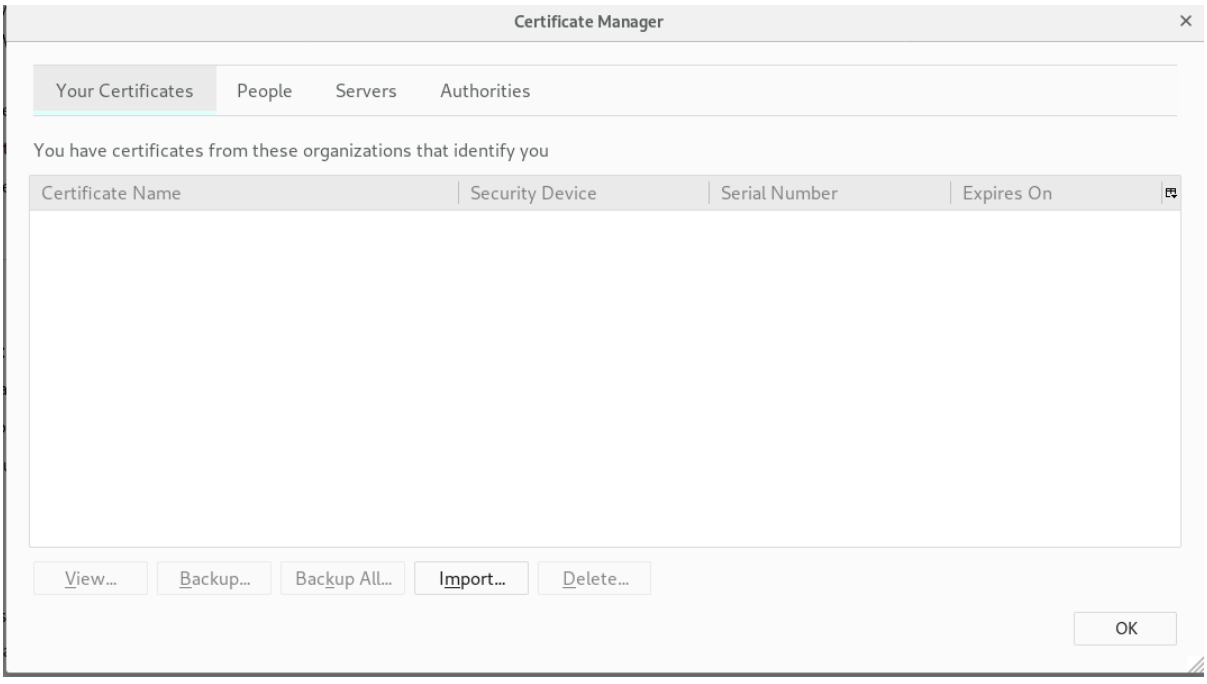
2. **Authorities** タブを選択し、**Import** ボタンをクリックします。



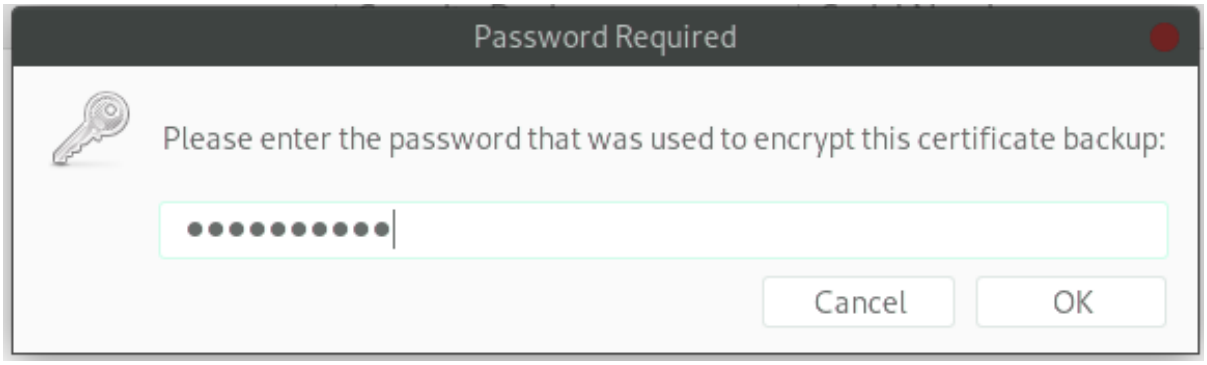
3. **ca.crt** ファイルを選択して、**Import** をクリックします。

クライアント証明書のインポート

1. **Options → Preferences → Privacy & Security→ View certificates** をクリックします。
2. **Your Certificates** タブを選択します。

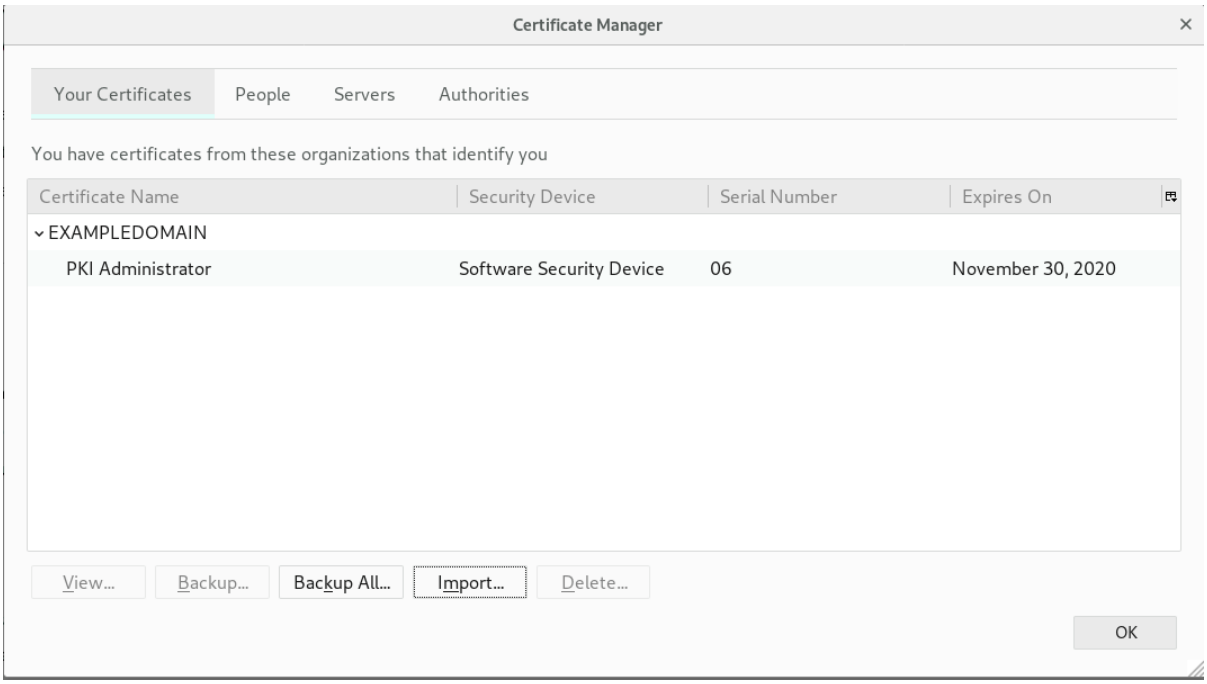


3. **Import** をクリックして、**ca_admin_cert.p12** などのクライアント p12 ファイルを選択します。
4. プロンプトにクライアント証明書のパスワードを入力します。



5. **OK** をクリックします。

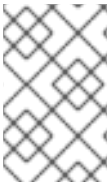
6. **Your Certificates** の下にエントリーが追加されていることを確認します。



Web コンソールへのアクセス
ブラウザーで **https://*host_name*:*ポート*** を開いて PKI サービスにアクセスできます。

2.4.2. 管理インターフェイス

すべてのサブシステムは HTML ベースの管理インターフェイスを使用します。ホスト名を入力し、URL としてセキュアなポートを入力し、管理者の証明書で認証し、適切な **Administrators** リンクをクリックします。



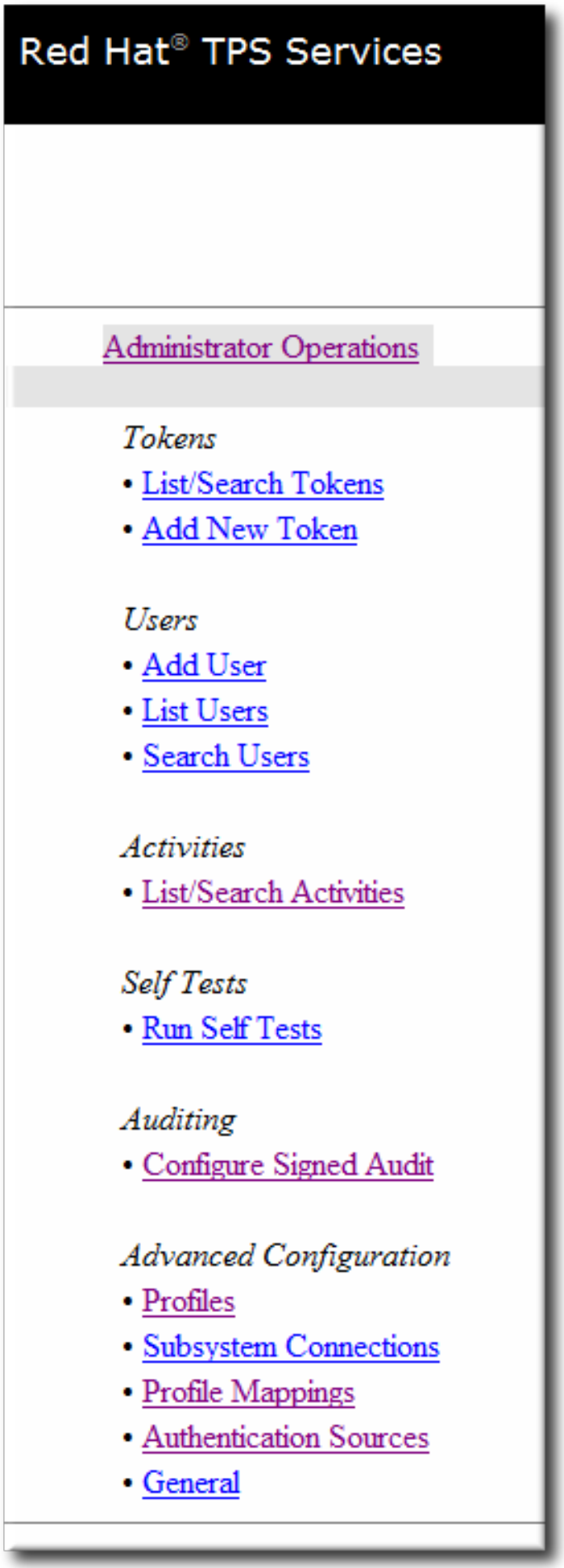
注記

管理者およびエージェントサービスの両方に使用されるすべてのサブシステムには、1 つの TLS ポートがあります。これらのサービスへのアクセスは、証明書ベースの認証により制限されます。

HTML 管理インターフェイスは Java コンソールよりもはるかに制限されています。プライマリー管理機能はサブシステムユーザーを管理します。

TPS では、操作は TPS サブシステムのユーザー管理のみを許可します。ただし、TPS 管理ページでは、トークンを一覧表示し、TPS で実行しているすべてのアクティビティー (通常は非表示管理アクションを含む) をすべて表示できます。

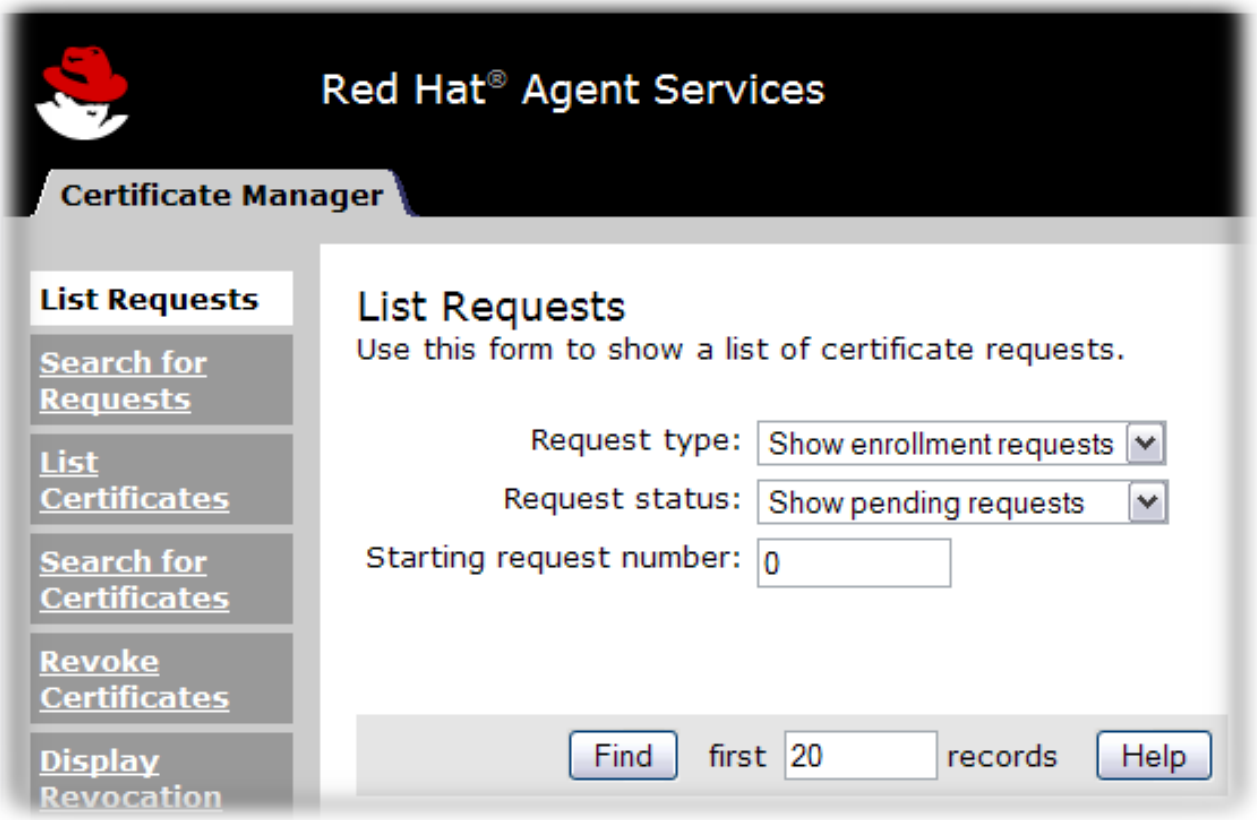
図2.2 TPS 管理ページ


Red Hat® TPS Services
Administrator Operations
Tokens
• List/Search Tokens
• Add New Token
Users
• Add User
• List Users
• Search Users
Activities
• List/Search Activities
Self Tests
• Run Self Tests
Auditing
• Configure Signed Audit
Advanced Configuration
• Profiles
• Subsystem Connections
• Profile Mappings
• Authentication Sources
• General

2.4.3. エージェントインターフェイス

エージェントサービスページは、証明書およびトークン管理タスクがほぼすべて実行されます。これらのサービスは HTML ベースのもので、エージェントは特別なエージェント証明書を使用してサイトに対して認証されます。

図2.3 Certificate Manager のエージェントサービスページ



操作はサブシステムによって異なります。

- Certificate Manager エージェントサービスには、(証明書を発行する) 証明書要求の承認、証明書の失効、ならびに証明書および CRL の公開が含まれます。CA が発行するすべての証明書は、そのエージェントサービスページで管理できます。
- TPS エージェントサービスは、CA エージェントサービスと同様、フォーマットされ、TPS を介して証明書が発行されたすべてのトークンを管理します。トークンはエージェントで登録、一時停止、および削除できます。他の 2 つのロール (operator および admin) は Web サービスページでトークンを表示できますが、トークンに対するアクションを実行できません。
- KRA エージェントサービスページは、キーリカバリー要求を処理します。キーリカバリー要求は、証明書が失われた場合に既存のキーペアを再利用して証明書を発行できるようにするかどうかを設定します。
- OCSP エージェントサービスページを使用すると、エージェントは CRL を OCSP に公開し、CRL を手動で OCSP に読み込み、クライアント OCSP 要求の状態を表示するように CA を設定します。

TKS は、エージェントサービスページのない唯一のサブシステムです。

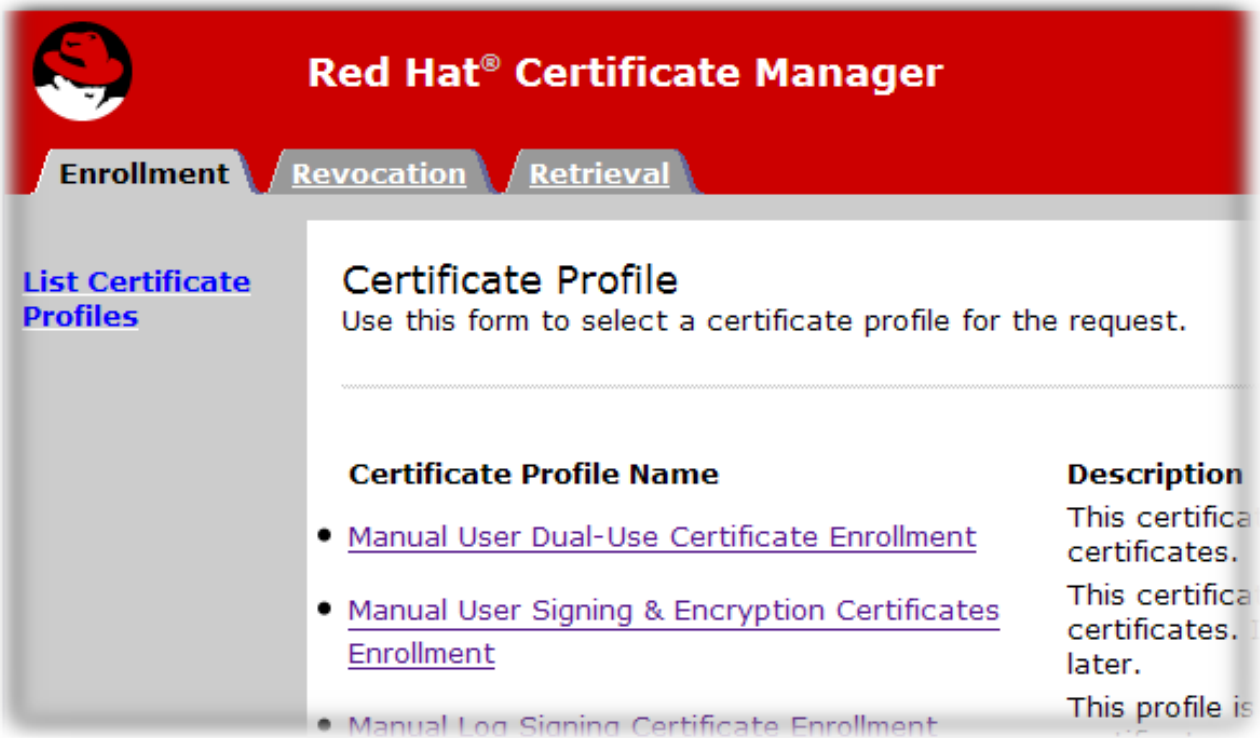
2.4.4. エンドユーザーページ

CA と TPS の両方は、ある方法で直接ユーザー要求を処理します。つまり、エンドユーザーにはこれらのサブシステムに接続する方法が必要です。CA にはエンドユーザーまたは **エンドエンティティー** の HTML サービスがあります。TPS は、Enterprise Security Client を使用します。

エンドユーザーサービスは、サーバーのホスト名と標準のポート番号を使用して標準の HTTP 経由でアクセスします。また、サーバーのホスト名および特定のエンドエンティティー TLS ポートを使用して、HTTPS 経由でもアクセスできます。

CA の場合、各タイプの TLS 証明書は、**プロファイル** と呼ばれる特定のオンライン送信フォームで処理されます。CA には約 20 ダースの証明書プロファイルがあり、証明書の種類 (ユーザー TLS 証明書、サーバー TLS 証明書、ログおよびファイル署名証明書、電子メール証明書、電子メール証明書、およびあらゆる種類のサブシステム証明書) に対応しています。カスタムプロファイルもあります。

図2.4 Certificate Manager のエンドエンティティーページ



エンドユーザーは、証明書の発行時に CA ページから証明書を取得します。また、CA チェーンと CRL をダウンロードし、それらのページから証明書を取り消したり更新したりすることもできます。

2.5. コマンドラインインターフェイス

本セクションでは、コマンドラインユーティリティーを説明します。

2.5.1. pkiCLI

pki コマンドラインインターフェイス (CLI) は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『REST インターフェイス』 を参照してください。CLI は以下のように呼び出すことができます。

```
$ pki [CLI options] <command> [command parameters]
```

CLI オプションは、コマンドの前に配置する必要があり、コマンドの後にコマンドパラメーターを指定する必要があることに注意してください。

2.5.1.1. pki CLI の初期化

コマンドラインインターフェイスを初めて使用するには、新しいパスワードを指定し、以下のコマンドを使用します。

```
$ pki -c <password> client-init
```

これにより、~/**.dogtag/nssdb** ディレクトリーに新しいクライアント NSS データベースが作成されます。パスワードは、クライアントの NSS データベースを使用するすべての CLI 操作に指定する必要があります。または、パスワードがファイルに保存されている場合は、**-C** オプションを使用してファイルを指定できます。以下に例を示します。

```
$ pki -C password_file client-init
```

クライアントの NSS データベースに CA 証明書をインポートするには、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『NSS データベースへの証明書のインポート』セクションを参照してください。

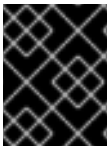
コマンドによっては、クライアント証明書の認証が必要な場合があります。既存のクライアント証明書とその鍵をクライアント NSS データベースにインポートするには、PKCS #12 ファイルとパスワードを指定して、以下のコマンドを実行します。

以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

```
$ openssl pkcs12 -in file -clcerts -nodes -nokeys -out file.crt
```

『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『証明書/キー暗号化トークンの管理』セクションに記載の手順に従って、管理クライアント証明書の検証およびインポートを行います。

```
$ PKICertImport -d ~/.dogtag/nssdb -n "nickname" -t ",," -a -i file.crt -u C
```



重要

CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

既存のクライアント証明書とその鍵をクライアント NSS データベースにインポートするには、PKCS #12 ファイルとパスワードを指定して、以下のコマンドを実行します。

```
$ pki -c <password> pkcs12-import --pkcs12-file <file> --pkcs12-password <password>
```

以下のコマンドを使用して、クライアント証明書を確認します。

```
certutil -V -u C -n "nickname" -d ~/.dogtag/nssdb
```

2.5.1.2. pki CLI の使用

コマンドラインインターフェイスは、階層構造で多数のコマンドをサポートします。トップレベルのコマンドを一覧表示するには、追加のコマンドまたはパラメーターを指定せずに **pki** コマンドを実行します。

```
$ pki
```

コマンドにはサブコマンドがあります。一覧を表示するには、コマンド名を指定して **pki** を実行して追加のオプションを指定せずに実行します。以下に例を示します。

```
$ pki ca
```

```
$ pki ca-cert
```

コマンドの使用情報を表示するには **--help** オプションを使用します。

```
$ pki --help
```

```
$ pki ca-cert-find --help
```

man ページを表示するには、コマンドラインの **help** コマンドを指定します。

```
$ pki help
```

```
$ pki help ca-cert-find
```

認証を必要としないコマンドを実行するには、コマンドとそのパラメーター (必要な場合) を指定します。以下に例を示します。

```
$ pki ca-cert-find
```

クライアント証明書の認証を必要とするコマンドを実行するには、証明書のニックネーム、クライアント NSS データベースのパスワード、および任意のサーバーの URL を指定します。

```
$ pki -U <server URL> -n <nickname> -c <password> <command> [command parameters]
```

以下に例を示します。

```
$ pki -n jsmith -c password ca-user-find ...
```

デフォルトでは、CLI は **http://*local_host_name*:8080** でサーバーと通信します。別の場所でサーバーと通信するには、URL を **-U** オプションで指定します。以下に例を示します。

```
$ pki -U https://server.example.com:8443 -n jsmith -c password ca-user-find
```

2.5.2. AtoB

AtoB ユーティリティーは、Base64 でエンコードされた証明書を、同等のバイナリーにデコードします。以下に例を示します。

```
$ AtoB input.ascii output.bin
```

詳細情報、その他のオプション、およびその他の例は、AtoB(1) の man ページを参照してください。

2.5.3. AuditVerify

AuditVerify ユーティリティーは、ログエントリーの署名を検証して、監査ログの整合性を検証します。

たとえば、以下のようになります。

```
$ AuditVerify -d ~jsmith/auditVerifyDir -n Log Signing Certificate -a ~jsmith/auditVerifyDir/logListFile -P "" -v
```

この例では、**~jsmith/auditVerifyDir** NSS データベース (**-d**) の **Log Signing Certificate** (**-n**) を使用して監査ログを検証します。検証するログのリスト (**-a**) は **~jsmith/auditVerifyDir/logListFile** ファイルにあります。こちらは、コンマ区切りで時系列順に並べられています。証明書の先頭に接頭辞 (**-P**) を追加し、キーデータベースのファイル名を空にします。出力は詳細です (**-v**)。

詳細、その他のオプション、およびその他の例は、AuditVerify(1) の man ページまたは「署名監査ログの使用」を参照してください。

2.5.4. BtoA

BtoA ユーティリティーは、バイナリーデータを Base64 でエンコードします。以下に例を示します。

```
$ BtoA input.bin output.ascii
```

詳細情報、その他のオプション、およびその他の例は、BtoA(1) の man ページを参照してください。

2.5.5. CMCRequest

CMCRequest ユーティリティーは、証明書の発行または失効要求を作成します。以下に例を示します。

```
$ CMCRequest example.cfg
```

注記

CMCRequest ユーティリティーのすべてのオプションは、ユーティリティーに渡される設定ファイルの一部として指定されます。設定ファイルのオプションと詳細は、CMCRequest(1) の man ページを参照してください。4.3 も参照してください。CMC および 「**CMCRequest** を使用した証明書の取り消し」 を使用した証明書の要求と受け取り

2.5.6. CMCRevoke

レガシー。使用しないでください。

2.5.7. CMCSharedToken

CMCSharedToken ユーティリティーは、共有秘密の CMC リクエストのユーザーパスフレーズを暗号化します。以下に例を示します。

```
$ CMCSharedToken -d . -p myNSSPassword -s "shared_passphrase" -o cmcSharedTok2.b64 -n "subsystemCert cert-pki-tomcat"
```

共有パスフレーズ (**-s**) は、現在のディレクトリー (**-d**) にある NSS データベースにある **subsystemCert cert-pki-tomcat** (**-n**) という名前の証明書を使用して、**cmcSharedtok2.b64** ファイル (**-o**) に暗号化されて保存されます。デフォルトのセキュリティートークン **internal** が使用され (**-h** が指

定されていないため)、トークンへのアクセスには **myNSSPassword** のトークンパスワードが使用されます。

詳細、その他のオプション、およびその他の例は、CMCSharedtoken(1) の man ページまたは「**CMCRequest** を使用した証明書の取り消し」を参照してください。

2.5.8. CRMFPopClient

CRMFPopClient ユーティリティーは、NSS データベースを使用する Certificate Request Message Format (CRMF) クライアントであり、Possession の Proof を提供します。

たとえば、以下のようになります。

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b kra.transport -w
"AES/CBC/PKCS5Padding" -t false -v -o /user_or_entity_database_directory/example.csr
```

この例では、**cn=subject_name** サブジェクト DN (**-n**)、現在のディレクトリー (**-d**) の NSS データベース (-b)、トランスポート (**kra.transport**) (**-b**) に使用する証明書、**AES/CBC/PKCS5Padding** キーをラップする証明書で新しい CSR を作成します (**-v**)。また、生成される CSR が **/user_or_entity_database_directory/example.csr** ファイル (**-o**) に書き込まれます。

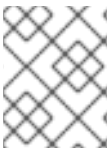
詳細やその他のオプション、追加の例は、**CRMFPopClient --help** コマンドの出力を参照してください。また、「**CMCRequest** を使用した証明書の取り消し」 も参照してください。

2.5.9. HttpClient

HttpClient ユーティリティーは、CMC 要求を送信するための NSS 対応の HTTP クライアントです。

たとえば、以下のようになります。

```
$ HttpClient request.cfg
```



注記

HttpClient ユーティリティーへのすべてのパラメーターは **request.cfg** ファイルに保存されます。詳細は、**HttpClient --help** コマンドの出力を参照してください。

2.5.10. OCSPClient

証明書失効リストのステータスを確認する Online Certificate Status Protocol (OCSP) クライアント。

たとえば、以下のようになります。

```
$ OCSPClient -h server.example.com -p 8080 -d /etc/pki/pki-tomcat/alias -c "caSigningCert cert-pki-
ca" --serial 2
```

この例では、ポート **8080** (**-p**) の **server.example.com** OCSP サーバー (**-h**) に対してクエリーを実行し、シリアル番号 **2** (**--serial**) を持つ **caSigningcet cert-pki-ca** (**-c**) が署名した証明書を確認します。**/etc/pki/pki-tomcat/alias** ディレクトリーの NSS データベースが使用されます。

詳細情報、その他のオプション、およびその他の例は、**OCSPClient --help** コマンドの出力を参照してください。

2.5.11. PKCS10Client

PKCS10Client ユーティリティーは、必要に応じて HSM 上に RSA キーおよび EC キーの CSR を PKCS10 形式で作成します。

たとえば、以下のようになります。

```
$ PKCS10Client -d /etc/dirsrv/slapd-instance_name/ -p password -a rsa -l 2048 -o ~/ds.csr -n
"CN=$HOSTNAME"
```

この例では、**/etc/dirsrv/slapd-instance_name/** ディレクトリー (**-d**) に 2048 ビット (**-l**) で、新しい RSA (**-a**) キーを、データベースパスワード (**password**) (**-p**) で作成します。出力 CSR は **~/ds.cfg** ファイル (**-o**) に格納されます。また、証明書 DN は **CN=$HOSTNAME** (**-n**) です。

詳細情報、その他のオプション、およびその他の例は、PKCS10Client(1) の man ページを参照してください。

2.5.12. PrettyPrintCert

PrettyPrintCert ユーティリティーは、人間が判読可能な形式で証明書の内容を表示します。

たとえば、以下のようになります。

```
$ PrettyPrintCert ascii_data.cert
```

このコマンドは、**ascii_data.cert** ファイルの出力を解析し、その内容を人間が読める形式で表示します。出力には、署名アルゴリズム、指数、モジュール、証明書拡張などの情報が含まれます。

詳細情報、その他のオプション、およびその他の例は、PrettyPrintCert(1) の man ページを参照してください。

2.5.13. PrettyPrintCrl

PrettyPrintCrl ユーティリティーは、CRL ファイルの内容を人間が判読できる形式で表示します。

たとえば、以下のようになります。

```
$ PrettyPrintCrl ascii_data.crl
```

このコマンドは、**ascii_data.crt** ファイルの出力を解析して、その内容を人間が読める形式で表示します。出力には、失効署名アルゴリズム、失効の発行者、取り消された証明書とその理由が一覧になったものなどが含まれます。

詳細情報、その他のオプション、およびその他の例は、PrettyPrintCrl(1) の man ページを参照してください。

2.5.14. TokenInfo

TokenInfo ユーティリティーは、NSS データベース内のトークンを一覧表示します。

たとえば、以下のようになります。

```
$ TokenInfo ./nssdb/
```

このコマンドは、指定のデータベースディレクトリーに登録されたすべてのトークン (HSM、ソフトトークンなど) を表示します。

詳細情報、その他のオプション、およびその他の例は、**TokenInfo** コマンドの出力を参照してください。

2.5.15. tkstool

tkstool ユーティリティーは、トークンキーサービス (TKS) サブシステムと対話します。

たとえば、以下のようになります。

```
$ tkstool -M -n new_master -d /var/lib/pki/pki-tomcat/alias -h token_name
```

このコマンドは、HSM **token_name** の **/var/lib/pki/pki-tomcat/alias** NSS データベースに **new_master** (**-n**) という名前の新しいマスターキー (**-M**) を作成します。

詳細情報、その他のオプション、およびその他の例は、**tkstool -H** コマンドの出力を参照してください。

2.6. ENTERPRISE SECURITY CLIENT

Enterprise Security Client は、スマートカードの管理を簡素化する Red Hat Certificate System のツールです。エンドユーザーは、セキュリティートークン (スマートカード) を使用して、シングルサインオンアクセスやクライアント認証などのアプリケーションのユーザー証明書を保存できます。エンドユーザーには、署名、暗号化、およびその他の暗号化機能に必要な証明書および鍵が含まれるトークンが発行されます。

Enterprise Security Client は、Certificate System の完全なトークン管理システムの 3 番目の部分です。2 つのサブシステム (Token Key Service(TKS) および Token Processing System(TPS)) は、トークン関連の操作を処理するために使用されます。**Enterprise Security Client** は、スマートカードとユーザーがトークン管理システムにアクセスできるようにするインターフェイスです。

トークンの登録後、Mozilla Firefox や Thunderbird などのアプリケーションは、トークンを認識して、クライアント認証や S/MIME メールなどのセキュリティー操作に使用するように設定できます。Enterprise Security Client は、以下の機能を提供します。

- Safenet の 330J スマートカードなどの JavaCard 2.1 以降のカードと Global Platform 2.01 準拠のスマートカードをサポートします。
- スマートカードと GemPCKey USB フォームファクターキーの両方である Gemalto TOP IM FIPS CY2 トークンをサポートします。
- SafeNet Smart Card 650 (SC650) をサポートします。
- セキュリティートークンを登録して、TPS で認識されるようにします。
- TPS でトークンを再登録するなど、セキュリティートークンを維持します。
- 管理対象トークンの現在のステータスに関する情報を提供します。
- トークンが失われた場合に別のトークンで鍵をアーカイブおよび復元できるように、サーバー側の鍵生成をサポートします。

Enterprise Security Client は、エンドユーザーがスマートカードまたはトークンで鍵と証明書を登録および管理するためのクライアントです。これは、Certificate System トークン管理システムの最終コンポーネントで、Token Processing System (TPS) および Token Key Service (TKS) です。

Enterprise Security Client は、トークン管理システムのユーザーインターフェイスを提供します。エンドユーザーは、署名、暗号化、およびその他の暗号化機能に必要な証明書および鍵が含まれるセキュリティートークンを発行できます。トークンを使用するには、TPS がトークンを認識して通信できるようにする必要があります。Enterprise Security Client は、トークンを登録する方法です。

Enterprise Security Client は、SSL/TLS HTTP チャンネル上で TPS のバックエンドと通信します。これは、ユーザーインターフェイスで拡張可能な Mozilla XULRunner フレームワークに基づいていますが、単純な HTML ベースの UI ではレガシー Web ブラウザーコンテナーを維持します。

トークンを適切に登録した後、Web ブラウザーはトークンを認識し、セキュリティー操作に使用するように設定できます。Enterprise Security Client は、以下の機能を提供します。

- ユーザーがセキュリティートークンを登録して、TPS によって認識されるようにします。
- ユーザーがセキュリティートークンを管理できるようにします。たとえば、Enterprise Security Client を使用すると、TPS でトークンを再登録できます。
- デフォルトおよびカスタムトークンプロファイルを使用したさまざまなトークンのサポートを提供します。デフォルトでは、TPS はユーザーキー、デバイスキー、およびセキュリティー担当者キーを自動的に登録できます。これにより、適切なプロファイルに従って (トークン CUID などの属性によって認識される) さまざまな使用に大してトークンが自動的に自動的に登録されるように、追加のプロファイルを追加できます。
- 管理対象トークンの現在のステータスに関する情報を提供します。

パート II. 証明書サービスの設定

第3章 証明書を発行するルール (証明書プロファイル) の作成

Certificate System は、受信証明書要求にポリシーを適用し、入力要求タイプと出力証明書タイプを制御するためのカスタマイズ可能なフレームワークを提供します。これらは **証明書プロファイル** と呼ばれます。証明書プロファイルは、Certificate Manager のエンドエンティティーページで証明書登録フォームに必要な情報を設定します。本章では、証明書プロファイルの設定方法を説明します。

3.1. 証明書プロファイルの概要

証明書プロファイルは、認証方法、認可方法、証明書のデフォルトの内容、内容の値の制約、証明書プロファイルの入力と出力の内容など、特定の種類の証明書の発行に関連するすべてを定義します。登録要求および更新要求は証明書プロファイルに送信され、その証明書プロファイルで設定されたデフォルトと制約の対象となります。これらの制約は、要求が証明書プロファイルに関連付けられた入力フォームを介して送信されるか、他の手段を介して送信されるかに関係なく適用されます。証明書プロファイル要求から発行される証明書には、デフォルトで必要なコンテンツと、デフォルトのパラメーターで必要な情報が含まれています。制約は、証明書で許可されるコンテンツに対するルールを提供します。

証明書プロファイルの使用およびカスタマイズの詳細は、「証明書プロファイルの設定」 を参照してください。

Certificate System には、デフォルトのプロファイルのセットが含まれています。デフォルトのプロファイルは、ほとんどのデプロイメントを満たすために作成されますが、すべてのデプロイメントは独自の新規証明書プロファイルを追加するか、既存のプロファイルを変更することができます。

- **認証。** すべての認証プロファイルで認証方法を指定できます。
- **認可。** すべての認定プロファイルで承認方法を指定できます。
- **プロファイル入力。** プロファイルの入力は、証明書が要求されたときに CA に送信されるパラメーターおよび値です。プロファイル入力には、証明書要求の公開鍵と、証明書の終了エンティティーによって要求される証明書のサブジェクト名が含まれます。
- **プロファイルの出力。** プロファイルの出力は、エンドエンティティーに証明書を提供する形式を指定するパラメーターおよび値です。プロファイルの出力は、要求が成功したときに PKCS#7 証明書チェーンが含まれる CMC の応答です。
- **証明書の内容。** 各証明書は、割り当てられたエンティティーの名前 (サブジェクト名)、署名アルゴリズム、有効期間などのコンテンツ情報を定義します。証明書に含まれるものは、X.509 標準で定義されます。X509 標準のバージョン 3 では、証明書に拡張機能を含めることもできます。証明書拡張の詳細は、??? を参照してください。

  証明書プロファイルに関する情報はすべて、プロファイルの設定ファイルのプロファイルポリシーの **set** エントリーで定義されます。複数の証明書が同時に要求される可能性がある場合は、各証明書のニーズを満たすためにプロファイルポリシーに複数のセットエントリーを定義できます。各ポリシーセットは多数のポリシールールで設定され、各ポリシールールは証明書コンテンツのフィールドを記述します。ポリシールールには、以下の内容を含めることができます。

  - **プロファイルのデフォルトです。** これらは、証明書内に含まれる情報に対する事前定義済みのパラメーターおよび許可される値です。プロファイルのデフォルトには、証明書の有効期間と、発行する証明書のタイプにどの証明書拡張が表示されるかが含まれます。
  - **プロファイルの制約。** 制約は、証明書を発行するルールまたはポリシーを設定します。また、プロファイル制約には、証明書のサブジェクト名に少なくとも 1 つの CN コンポーネントを含める必要があるルールが含まれます。また、証明書の有効性を最大 360 日に設定し

て、更新を許可する猶予期間を定義するルール、または **subjectaltname** 拡張が常に **true** に設定される必要があるというルールが含まれます。

3.1.1. 登録プロファイル

入力、出力、およびポリシーセットを定義する各プロファイルのパラメーターは、Table 11.1 に詳細に記載されています。Red Hat Certificate System 計画、インストールガイド、およびデプロイメントのガイドのプロファイル設定ファイルのパラメーター

プロファイルには、例3.1「caCMCUserCert プロファイルの例」 の **caUserCert** プロファイルで説明されているように、通常、入力、ポリシーセット、および出力が含まれます。

例3.1 caCMCUserCert プロファイルの例

証明書プロファイルの最初の部分は説明です。これは、名前、長い説明、有効かどうか、および有効であるかを表示します。

```
desc=This certificate profile is for enrolling user certificates by using the CMC certificate request with CMC Signature authentication.
visible=true
enable=true
enableBy=admin
name=Signed CMC-Authenticated User Certificate Enrollment
```

注記

このプロファイルにない **auth.instance_id=** エントリーは、このプロファイルを使用した登録リクエストの送信に認証は必要ありません。ただし、保証を取得するには、承認された CA エージェントによる手動承認が必要です。

次に、プロファイルはプロファイルに必要なすべての入力を一覧表示します。

```
input.list=i1
input.i1.class_id=cmcCertReqInputImp
```

caCMCUserCert プロファイルの場合、これは、証明書要求タイプ (CMC) を定義します。

次に、プロファイルは出力 (最終証明書の形式) を定義する必要があります。唯一利用できるのは **certOututIcmpl** で、成功すると、CMC 応答が要求元に戻ります。

```
output.list=o1
output.o1.class_id=certOutputImpl
```

最後の (最大の) 設定ブロックは、プロファイルに設定されたポリシーです。ポリシーは、有効期間、更新設定、証明書が使用できるアクションなど、最終的な証明書に適用されるすべての設定一覧を設定します。**policyset.list** パラメーターは、1 つの証明書に適用されるポリシーのブロック名を識別します。適用する個々のポリシーが **policyset.userCertSet.list** により一覧表示されます。

たとえば、6 番目のポリシーは、ポリシーの設定に従って、証明書に Key Usage Extension を自動的に入力します。これは、デフォルトを設定し、制約を設定して証明書でそれらのデフォルトを使用するようにする必要があります。

```
policyset.list=userCertSet
```

```
policyset.userCertSet.list=1,10,2,3,4,5,6,7,8,9
...
policyset.userCertSet.6.constraint.class_id=keyUsageExtConstraintImpl
policyset.userCertSet.6.constraint.name=Key Usage Extension Constraint
policyset.userCertSet.6.constraint.params.keyUsageCritical=true
policyset.userCertSet.6.constraint.params.keyUsageDigitalSignature=true
policyset.userCertSet.6.constraint.params.keyUsageNonRepudiation=true
policyset.userCertSet.6.constraint.params.keyUsageDataEncipherment=false
policyset.userCertSet.6.constraint.params.keyUsageKeyEncipherment=true
policyset.userCertSet.6.constraint.params.keyUsageKeyAgreement=false
policyset.userCertSet.6.constraint.params.keyUsageKeyCertSign=false
policyset.userCertSet.6.constraint.params.keyUsageCrlSign=false
policyset.userCertSet.6.constraint.params.keyUsageEncipherOnly=false
policyset.userCertSet.6.constraint.params.keyUsageDecipherOnly=false
policyset.userCertSet.6.default.class_id=keyUsageExtDefaultImpl
policyset.userCertSet.6.default.name=Key Usage Default
policyset.userCertSet.6.default.params.keyUsageCritical=true
policyset.userCertSet.6.default.params.keyUsageDigitalSignature=true
policyset.userCertSet.6.default.params.keyUsageNonRepudiation=true
policyset.userCertSet.6.default.params.keyUsageDataEncipherment=false
policyset.userCertSet.6.default.params.keyUsageKeyEncipherment=true
policyset.userCertSet.6.default.params.keyUsageKeyAgreement=false
policyset.userCertSet.6.default.params.keyUsageKeyCertSign=false
policyset.userCertSet.6.default.params.keyUsageCrlSign=false
policyset.userCertSet.6.default.params.keyUsageEncipherOnly=false
policyset.userCertSet.6.default.params.keyUsageDecipherOnly=false
...
```

3.1.2. 証明書拡張: デフォルトおよび制約

拡張機能は、証明書の使用方法に関する証明書またはルールに含める追加情報を設定します。これらの拡張機能は、証明書要求に指定するか、プロファイルのデフォルト定義から取得した後、制約によって適用できます。

証明書拡張機能がプロファイルで追加または識別されるには、拡張機能に対応する **デフォルト** を追加し、証明書拡張機能が要求で設定されていない場合にデフォルト値を設定します。たとえば、Basic Constraints Extension は、証明書が CA 署名証明書であるかどうか、CA の下で設定できる従属 CA の最大数、および拡張機能が重要 (必須) であるかどうかを識別します。

```
policyset.caCertSet.5.default.name=Basic Constraints Extension Default
policyset.caCertSet.5.default.params.basicConstraintsCritical=true
policyset.caCertSet.5.default.params.basicConstraintsIsCA=true
policyset.caCertSet.5.default.params.basicConstraintsPathLen=-1
```

拡張機能は、**constraints** と呼ばれる証明書要求に必要な値を設定することもできます。リクエストの内容がセット制約と一致しない場合、リクエストは拒否されます。制約は通常、拡張機能のデフォルトに対応しますが、常にそうとは限りません。以下に例を示します。

```
policyset.caCertSet.5.constraint.class_id=basicConstraintsExtConstraintImpl
policyset.caCertSet.5.constraint.name=Basic Constraint Extension Constraint
policyset.caCertSet.5.constraint.params.basicConstraintsCritical=true
policyset.caCertSet.5.constraint.params.basicConstraintsIsCA=true
policyset.caCertSet.5.constraint.params.basicConstraintsMinPathLen=-1
policyset.caCertSet.5.constraint.params.basicConstraintsMaxPathLen=-1
```

-

注記

ユーザーが指定する拡張機能を証明書要求に組み込むのを許可し、プロファイルでシステム定義のデフォルトを無視するには、プロファイルに、「User Supplied Extension Default」 で説明されているユーザー Supplied Extension Default を含める必要があります。

3.1.3. 入力および出力

証明書を受信するために送信する必要がある入力セット情報。これは、要求側の情報、証明書要求の特定の形式、または組織情報のいずれかになります。

プロファイルで設定された出力では、発行された証明書の形式を定義します。

Certificate System では、エンドエンティティーページを介してアクセスされる **登録フォーム** を使用して、ユーザーがプロファイルにアクセスします。TPS などのクライアントも、これらの形式を介して登録リクエストを送信します。 その後、入力は登録フォームのフィールドに対応します。この出力は、証明書取得ページに含まれる情報に対応します。

3.2. 証明書プロファイルの設定

証明書システムでは、登録プロファイルを追加、削除、および変更できます。

- PKI コマンドラインインターフェイスの使用
- Java ベースの管理コンソールの使用

本セクションでは、各メソッドに関する情報を提供します。

3.2.1. PKI コマンドラインインターフェイスを使用した証明書の登録プロファイルの管理

本セクションでは、**pki** ユーティリティーを使用して証明書プロファイルを管理する方法を説明します。詳細は、pki-ca-profile(1) の man ページを参照してください。

注記

RAW 形式の使用が推奨されます。プロファイルの各属性およびフィールドの詳細は、Red Hat Certificate System 計画、インストール、およびデプロイメントガイドの証明書プロファイルの作成および編集を参照してください。

3.2.1.1. 証明書プロファイルの有効化および無効化

証明書プロファイルを編集する前に、無効にする必要があります。変更が完了したら、プロファイルを再度有効にできます。

注記

CA エージェントのみが、証明書プロファイルを有効化および無効化できます。

たとえば、**caCMCECserverCert** 証明書プロファイルを無効にするには、次のコマンドを実行します。

```
# pki -c password -n caagent ca-profile-disable caCMCECserverCert
```

たとえば、**caCMCECserverCert** 証明書プロファイルを有効にするには、次のコマンドを実行します。

```
# pki -c password -n caagent ca-profile-enable caCMCECserverCert
```

3.2.1.2. Raw 形式の証明書プロファイルの作成

新規プロファイルを raw 形式で作成するには、次のコマンドを実行します。

```
# pki -c password -n caadmin ca-profile-add profile_name.cfg --raw
```

注記

raw 形式で、以下のように新しいプロファイル ID を指定します。

```
profileId=profile_name
```

3.2.1.3. RAW 形式での証明書プロファイルの編集

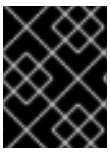
CA 管理者は、設定ファイルを手動でダウンロードせずに、RAW 形式で証明書プロファイルを編集できます。

たとえば、**caCMCECserverCert** プロファイルを編集するには、次のコマンドを実行します。

```
# pki -c password -n caadmin ca-profile-edit caCMCECserverCert
```

このコマンドは、プロファイル設定を RAW 形式で自動的にダウンロードし、**VI** エディターで開きます。エディターを閉じると、サーバーでプロファイル設定が更新されます。

プロファイルの編集後に CA を再起動する必要はありません。



重要

プロファイルを編集する前に、プロファイルを無効にします。詳細は、「証明書プロファイルの有効化および無効化」 を参照してください。

例3.2 RAW 形式での証明書プロファイルの編集

たとえば、**caCMCserverCert** プロファイルを編集して、ユーザーが提供する複数の拡張機能を許可するには、次を行います。

1. CA エージェントであるプロファイルを無効にします。

   ```
   # pki -c password -n caagemt ca-profile-disable caCMCserverCert
   ```

2. プロファイルを CA 管理者として編集します。

   a. **VI** エディターでプロファイルをダウンロードして開きます。

      ```
      # pki -c password -n caadmin ca-profile-edit caCMCserverCert
      ```

b. 設定を更新して、拡張機能を受け入れます。詳細は、??? を参照してください。

3. プロファイルを CA エージェントとして有効にします。

```
# pki -c password -n caagent ca-profile-enable caCMCserverCert
```

3.2.1.4. 証明書プロファイルの削除

証明書プロファイルを削除するには、次のコマンドを実行します。

```
# pki -c password -n caadmin ca-profile-del profile_name
```

重要

プロファイルを削除する前に、プロファイルを無効にします。詳細は、「証明書プロファイルの有効化および無効化」 を参照してください。

3.2.2. Java ベースの管理コンソールを使用した証明書の登録プロファイルの管理

重要

pkiconsole が非推奨になりました。

3.2.2.1. CA コンソールを使用した証明書プロファイルの作成

セキュリティー上の理由から、Certificate System は、ロールの分離を強制します。これにより、既存の証明書プロファイルは、エージェントによって許可された後にのみ管理者が編集できます。新しい証明書プロファイルを追加するか、既存の証明書プロファイルを変更するには、管理者として以下の手順を実施します。

1. Certificate System CA サブシステムコンソールにログインします。

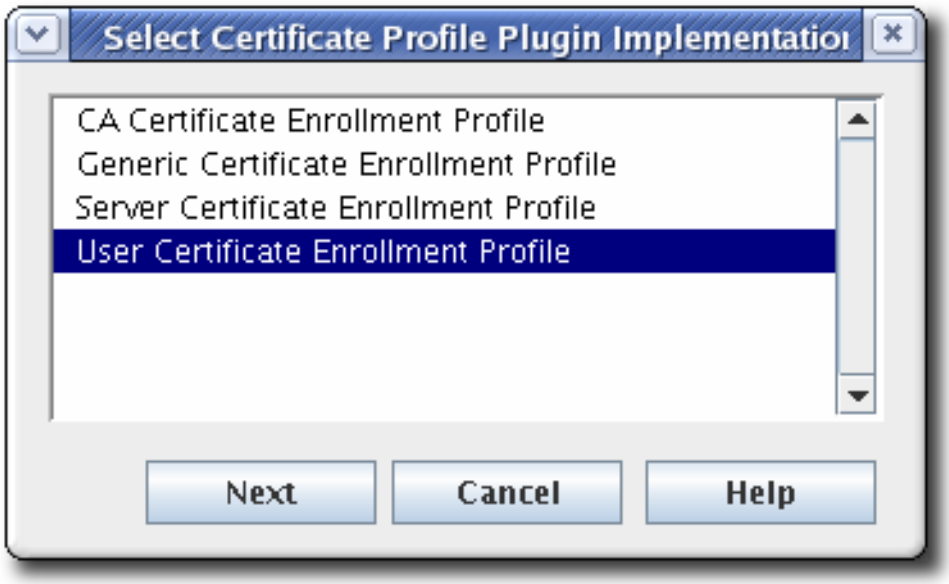
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで **Certificate Manager** を選択し、**Certificate Profiles** を選択します。

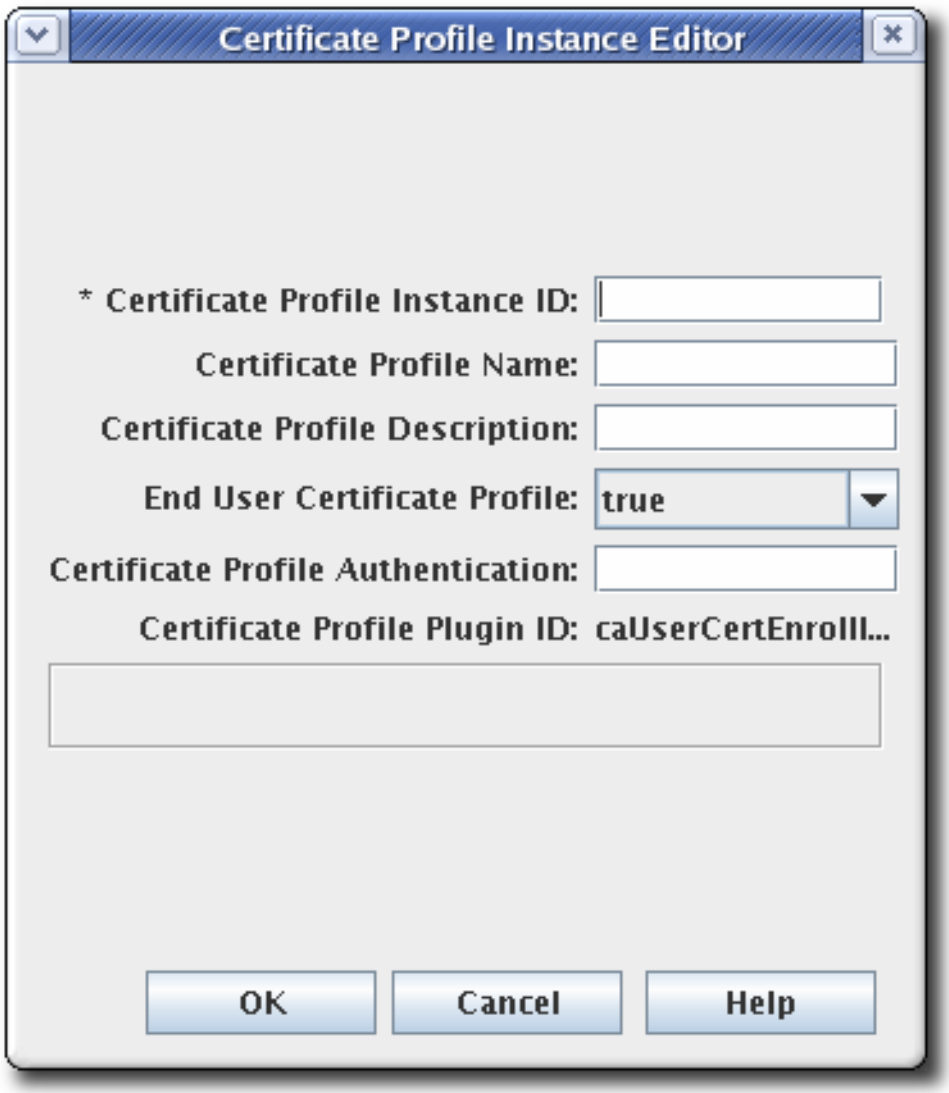
   設定した証明書プロファイルを一覧表示する **Certificate Profile Instances Management** タブが開きます。

3. 新しい証明書プロファイルを作成するには、**Add** をクリックします。

   Select Certificate Profile Plugin Implementation ウィンドウで、プロファイルが作成される証明書のタイプを選択します。



4. **Certificate Profile Instance Editor** にプロファイル情報を入力します。



- **Certificate Profile Instance ID**。この ID は、システムがプロファイルの特定に使用する ID です。
- **証明書プロファイル名**これは、ユーザーが分かりやすいプロファイルの名前です。

- **Certificate Profile Description**。

- **End User Certificate Profile**。これにより、リクエストがプロファイルの入力フォームを介して行われる必要があるかどうかが設定されます。これは通常 **true** に設定されます。これを **false** に設定すると、証明書プロファイルの入力ページではなく、Certificate Manager の証明書プロファイルフレームワークを介して署名済みリクエストが処理できるようになります。

- **証明書プロファイル認証**これにより、認証方法が設定されます。認証インスタンスのインスタンス ID を指定して、自動認証が設定されます。このフィールドが空の場合、認証方法はエージェントで承認される登録になります。要求はエージェントサービスインターフェイスの要求キューに送信されます。

  TMS サブシステム用でない限り、管理者は次の認証プラグインのいずれかを選択する必要があります。

  - **CMCAuth**: CA エージェントが登録要求を承認し、送信する必要がある場合に、このプラグインを使用します。

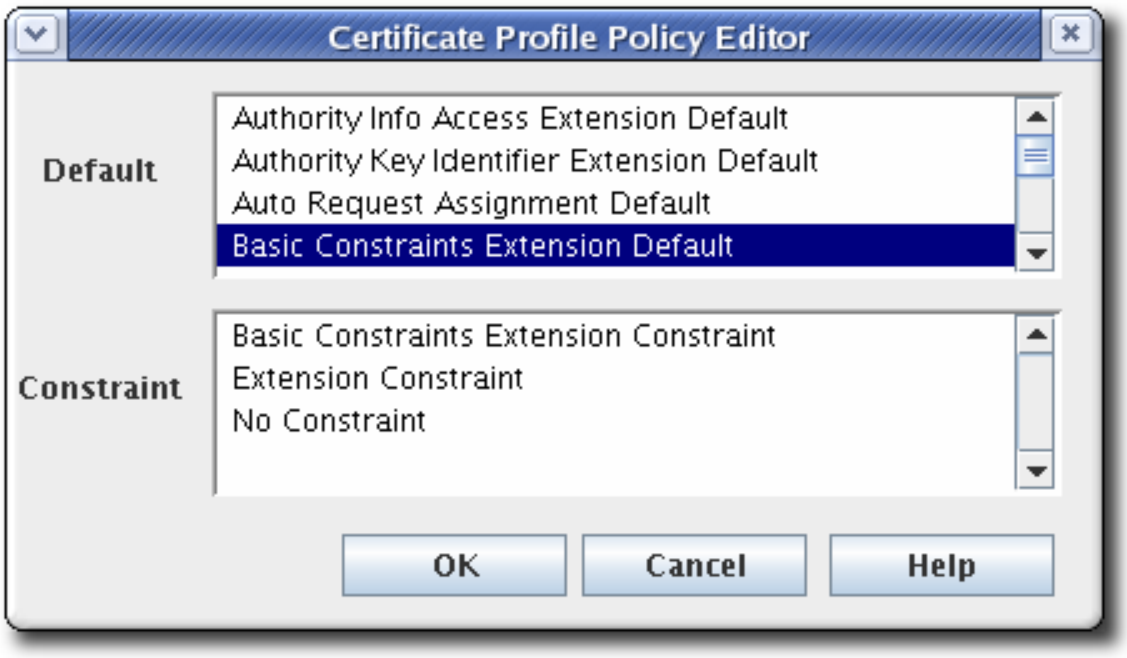
  - **CMCUserSignedAuth**: このプラグインを使用して、エージェント以外のユーザーが独自の証明書を登録できるようにします。

5. **OK** をクリックします。プラグインエディターが閉じられ、新しいプロファイルが profiles タブに一覧表示されます。

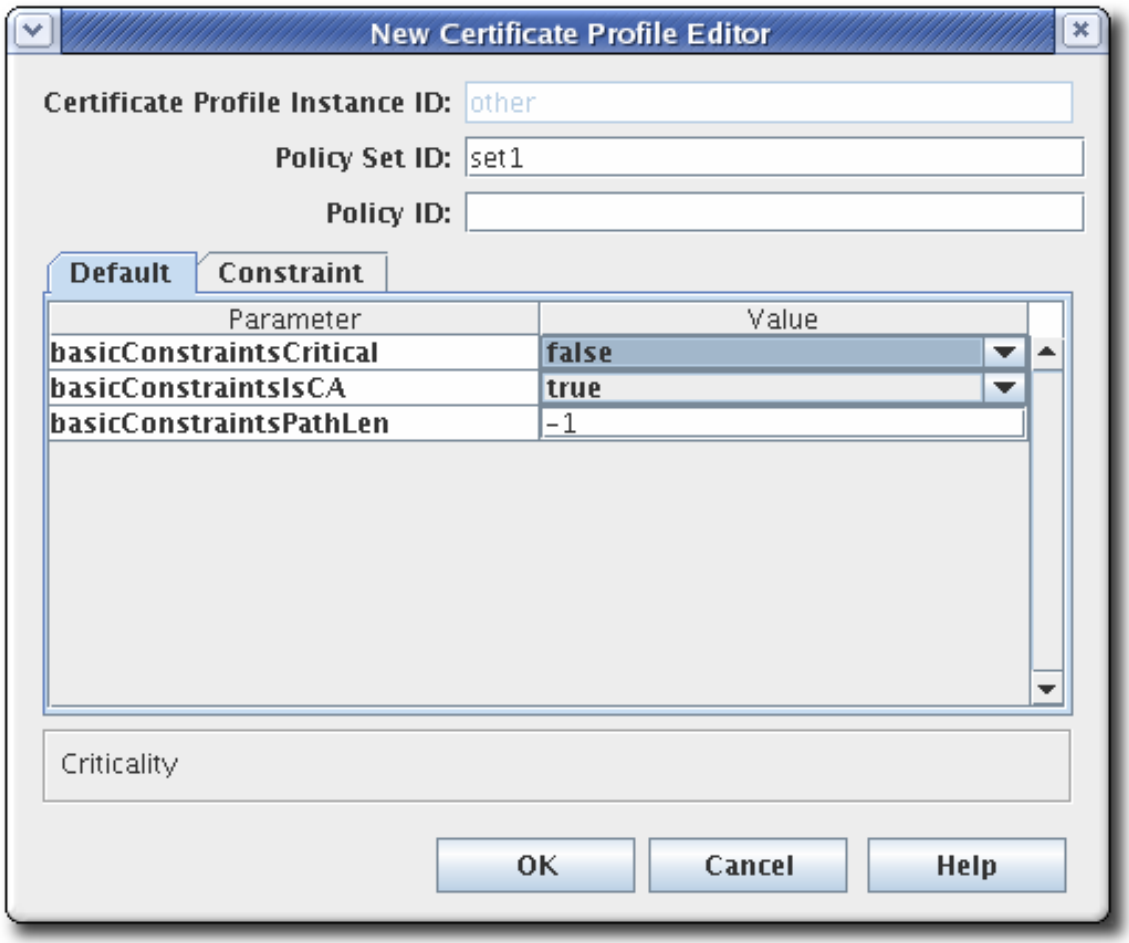
6. 新規プロファイルのポリシー、入力、および出力を設定します。一覧から新しいプロファイルを選択し、**Edit/View** をクリックします。

7. **Certificate Profile Rule Editor** ウィンドウの **Policies** タブでポリシーを設定します。**ポリシー** タブには、プロファイルタイプに対してデフォルトで設定されているポリシーが一覧表示されます。

   a. ポリシーを追加するには、**Add** をクリックします。

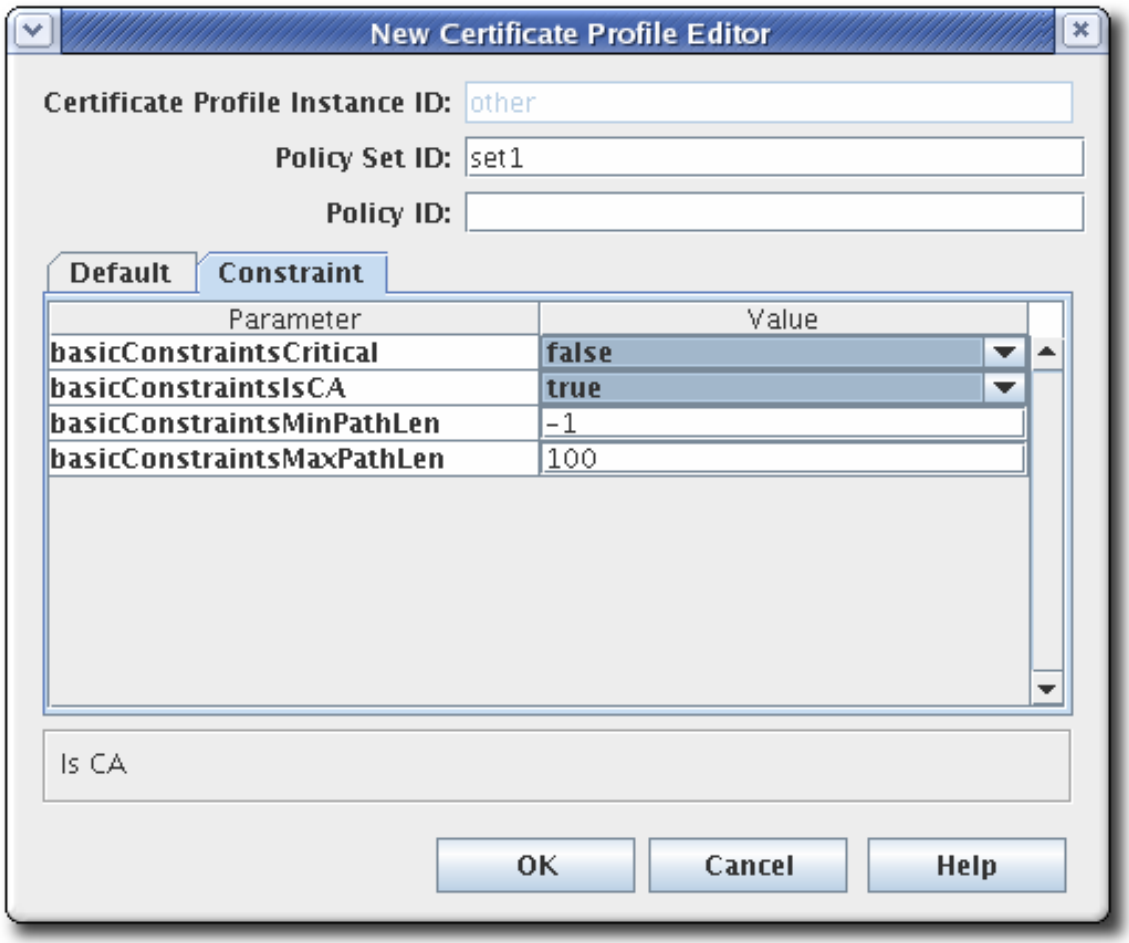


   b. **Default** フィールドからデフォルトを選択して、**Constraints** フィールドでそのポリシーに関連付けられた制約を選択し、**OK** をクリックします。



c. ポリシー設定 ID を入力します。デュアルキーペアを発行する場合には、個別のポリシーセットで、各証明書に関連付けられたポリシーを定義します。次に、証明書プロファイルポリシー ID と、証明書プロファイルポリシーの名前または識別子を入力します。

d. **Defaults** タブおよび **Constraints** タブでパラメーターを設定します。



Defaults は、証明書要求に設定する属性を定義します。これにより、証明書の内容が決定されます。これらは、拡張、有効期間、または証明書に含まれるその他のフィールドのいずれかになります。**制約** は、デフォルト値の有効な値を定義します。

各デフォルトまたは制約の詳細は、「デフォルトの参照」 および 「制約の参照」 を参照してください。

既存のポリシーを変更するには、ポリシーを選択し、**Edit** をクリックします。次に、そのポリシーのデフォルトおよび制約を編集します。

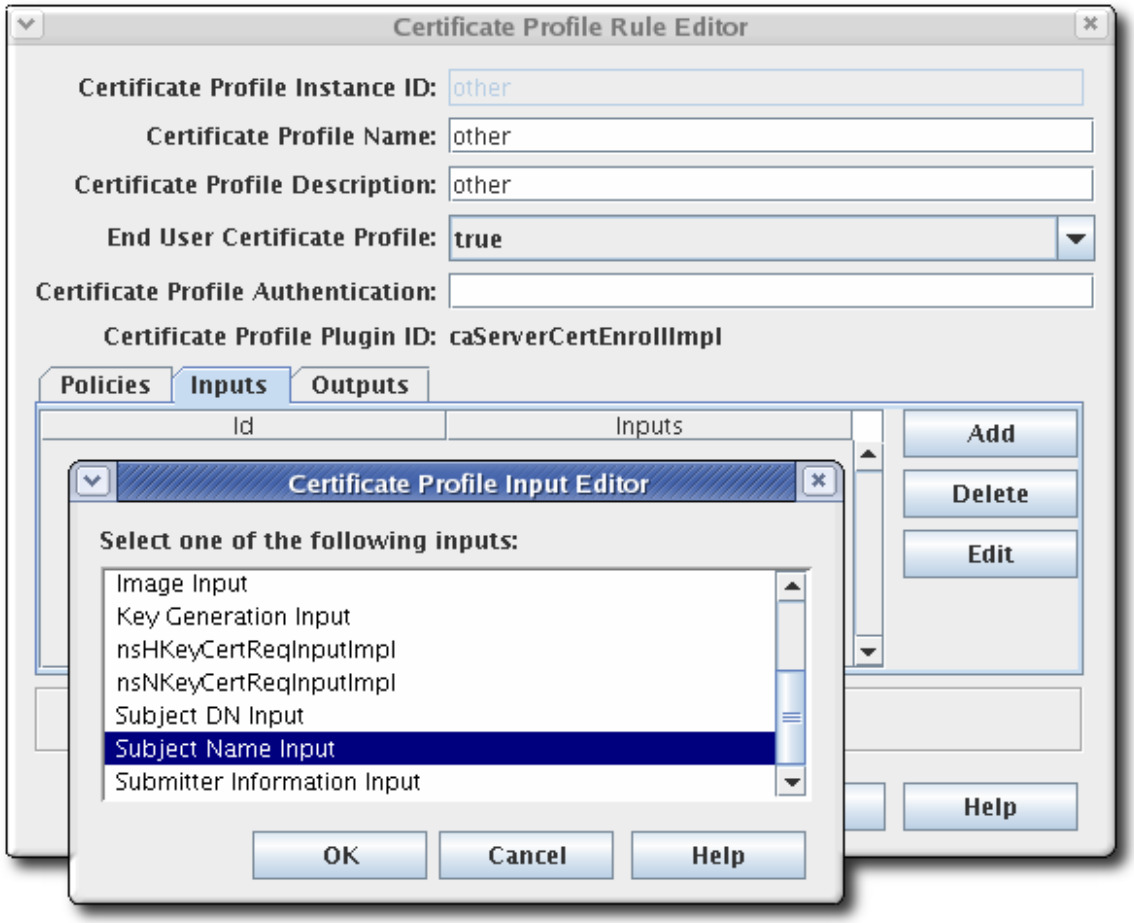
ポリシーを削除するには、ポリシーを選択し、**Delete** をクリックします。

8. **Certificate Profile Rule Editor** ウィンドウの **Inputs** タブに入力を設定します。プロファイルには複数の入力タイプが存在します。

注記

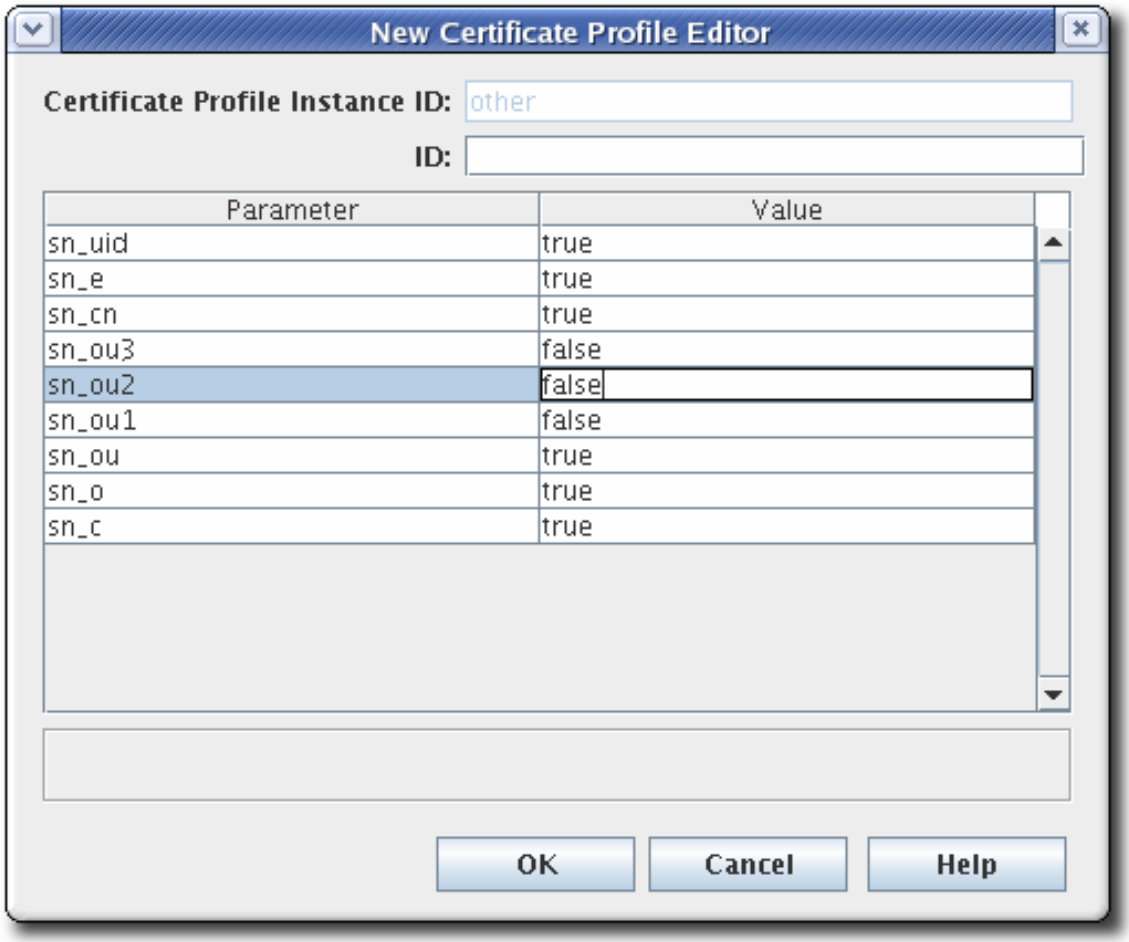
TMS サブシステムにプロファイルを設定しない場合は、**cmcCertReqInput** のみを選択して **Delete** ボタンをクリックし、他のプロファイルを削除します。

a. 入力を追加するには、**Add** をクリックします。



b. 一覧から入力を選択し、**OK** をクリックします。デフォルト入力の完全な詳細については、「入力の参照」 を参照してください。

c. **New Certificate Profile Editor** ウィンドウが開きます。入力 ID を設定して、**OK** をクリックします。



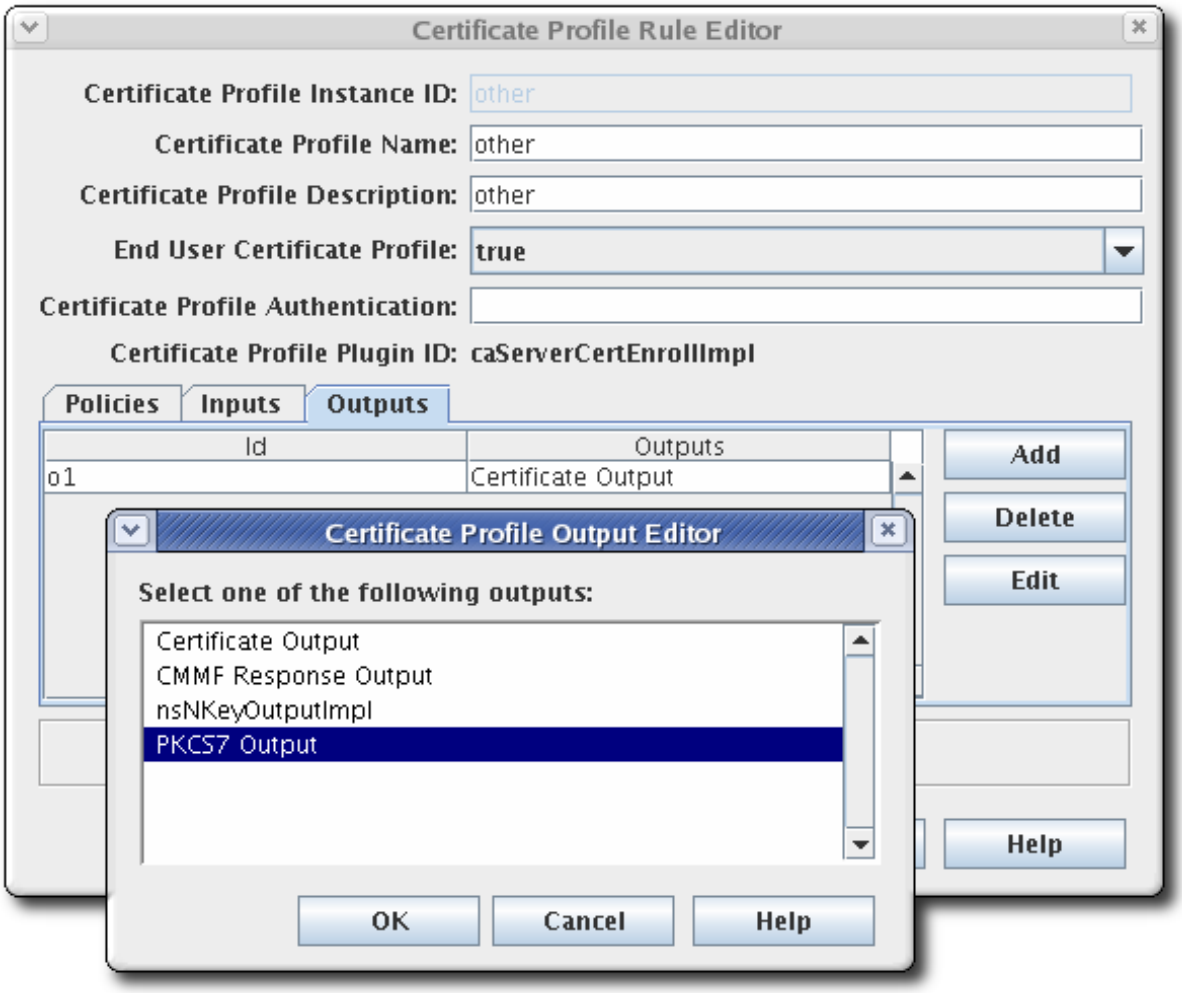
入力は追加および削除が可能です。入力の編集を選択することは可能ですが、入力にはパラメーターやその他の設定がないため、設定するものはありません。

入力を削除するには、入力を選択して **Delete** をクリックします。

9. **Certificate Profile Rule Editor** ウィンドウの **Outputs** タブで、出力を設定します。

   自動認証方式を使用する証明書プロファイルには、出力を設定する必要があります。エージェントが承認した認証を使用する証明書プロファイルには、出力を設定する必要はありません。Certificate Output タイプはすべてのプロファイルでデフォルトで設定され、カスタムプロファイルに自動的に追加されます。

   TMS サブシステムにプロファイルを設定しない限り、**certOutput** のみを選択します。



出力を追加または削除できます。出力の編集を選択することは可能ですが、出力にはパラメーターやその他の設定がないため、設定するものはありません。

a. 出力を追加するには、**Add** をクリックします。

b. 一覧から出力を選択して **OK** をクリックします。

c. 出力の名前または識別子を指定して、**OK** をクリックします。

この出力は出力タブに一覧表示されます。これを編集して、この出力のパラメーターに値を指定できます。

出力を削除するには、一覧から出力を選択して **Delete** をクリックします。

10. CA を再起動して、新規プロファイルを適用します。

```
systemctl restart pki-tomcatd-nuxwdog@instance_name.service
```

11. プロファイルを管理者として作成したら、CA エージェントはエージェントサービスページでプロファイルを承認してプロファイルを有効にする必要があります。

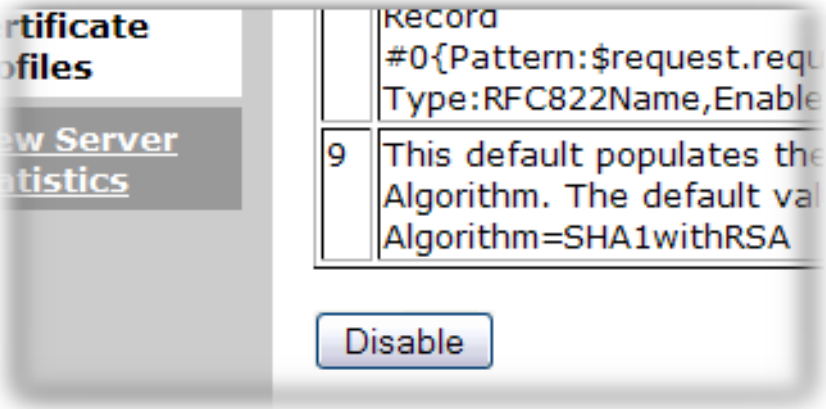
a. CA のサービスページを開きます。

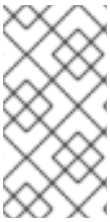
```
https://server.example.com:8443/ca/services
```

b. **証明書プロファイルの管理** リンクをクリックします。このページには、アクティブと非アクティブの両方で、管理者によって設定されたすべての証明書プロファイルが一覧表示されます。

c. 承認する証明書プロファイルの名前をクリックします。

d. ページの下部で、**Enable** ボタンをクリックします。





注記

このプロファイルを TPS で使用する場合は、プロファイルタイプを認識するように TPS を設定する必要があります。これは 11.1.4. にあります。Red Hat Certificate System の計画、インストール、およびデプロイメントガイドのスマートカード CA プロファイルの管理

プロファイルの承認方法は、Red Hat Certificate System の計画、インストール、およびデプロイメントガイドのファイルシステムの証明書プロファイルの作成および編集のセクションで説明されているように、コマンドラインでのみプロファイルに追加できます。

3.2.2.2. コンソールでの証明書プロファイルの編集

たとえば、既存の証明書プロファイルを修正するには、以下の手順に従います。

1. エージェントサービスページにログインし、プロファイルを無効にします。

   エージェントで証明書プロファイルを有効にすると、その証明書プロファイルは **Certificate Profile Instance Management** タブで有効とマークされ、コンソールを介して証明書プロファイルはいつでも編集できません。

2. Certificate System CA サブシステムコンソールにログインします。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

3. **Configuration** タブで **Certificate Manager** を選択し、**Certificate Profiles** を選択します。

4. 証明書のプロファイルを選択して、**Edit/View** をクリックします。

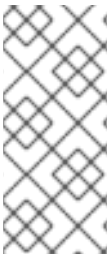
5. **Certificate Profile Rule Editor** ウィンドウが表示されます。多くのデフォルト、制約、入力、または出力が変更されています。

注記

プロファイルインスタンス ID は変更しないでください。

必要に応じて、ウィンドウの隅の 1 つを引っ張って、ウィンドウを拡大します。

6. CA を再起動して変更を適用します。
7. エージェントサービスページで、プロファイルを再度有効にします。



注記

エージェントによって承認されない証明書プロファイルを削除します。**Certificate Profile Instance Management** タブに表示される証明書プロファイルも、エージェントサービスインターフェイスに表示されます。プロファイルがすでに有効になっている場合は、プロファイルリストから削除する前に、エージェントによって無効にする必要があります。

3.2.3. 証明書の登録プロファイルの一覧表示

以下の事前定義済みの証明書プロファイルを使用し、Certificate System CA のインストール時にこの環境で使用できます。これらの証明書プロファイルは、最も一般的なタイプの証明書用に設計されており、一般的なデフォルト、制約、認証方法、入力、および出力を提供します。

コマンドラインで利用可能なプロファイルを一覧表示するには、**pki** ユーティリティーを使用します。以下に例を示します。

```
# pki -c password -n caadmin ca-profile-find
-----------------
59 entries matched
-----------------
  Profile ID: caCMCserverCert
  Name: Server Certificate Enrollment using CMC
  Description: This certificate profile is for enrolling server certificates using CMC.

  Profile ID: caCMCECserverCert
  Name: Server Certificate wth ECC keys Enrollment using CMC
  Description: This certificate profile is for enrolling server certificates with ECC keys using CMC.

  Profile ID: caCMCECsubsystemCert
  Name: Subsystem Certificate Enrollment with ECC keys using CMC
  Description: This certificate profile is for enrolling subsystem certificates with ECC keys using CMC.

  Profile ID: caCMCsubsystemCert
  Name: Subsystem Certificate Enrollment using CMC
  Description: This certificate profile is for enrolling subsystem certificates using CMC.

  ...
----------------------------
Number of entries returned 20
```

詳細は、pki-ca-profile(1) の man ページを参照してください。追加の情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントガイド』を参照してください。

3.2.4. 証明書登録プロファイルの詳細表示

たとえば、**caECFullCMCUserSignedCert** などの特定の証明書プロファイルを表示するには、次のコマンドを実行します。

```
$ pki -c password -n caadmin ca-profile-show caECFullCMCUserSignedCert
-----------------------------------
Profile "caECFullCMCUserSignedCert"
-----------------------------------
  Profile ID: caECFullCMCUserSignedCert
  Name: User-Signed CMC-Authenticated User Certificate Enrollment
  Description: This certificate profile is for enrolling user certificates with EC keys by using the CMC
certificate request with non-agent user CMC authentication.

  Name: Certificate Request Input
  Class: cmcCertReqInputImpl

    Attribute Name: cert_request
    Attribute Description: Certificate Request
    Attribute Syntax: cert_request

  Name: Certificate Output
  Class: certOutputImpl

    Attribute Name: pretty_cert
    Attribute Description: Certificate Pretty Print
    Attribute Syntax: pretty_print

    Attribute Name: b64_cert
    Attribute Description: Certificate Base-64 Encoded
    Attribute Syntax: pretty_print
```

たとえば、**caECFullCMCUserSignedCert** などの特定の証明書プロファイルを表示するには、raw 形式で次のコマンドを実行します。

```
$ pki -c password -n caadmin ca-profile-show caECFullCMCUserSignedCert --raw
#Wed Jul 25 14:41:35 PDT 2018
auth.instance_id=CMCUserSignedAuth
policyset.cmcUserCertSet.1.default.params.name=
policyset.cmcUserCertSet.4.default.class_id=authorityKeyIdentifierExtDefaultImpl
policyset.cmcUserCertSet.6.default.params.keyUsageKeyCertSign=false
policyset.cmcUserCertSet.10.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.10.constraint.name=Renewal Grace Period Constraint
output.o1.class_id=certOutputImpl

...
```

詳細は、pki-ca-profile(1) の man ページを参照してください。

3.3. プロファイルでの鍵のデフォルトの定義

証明書プロファイルの作成時に **サブジェクトキー識別子のデフォルトの前にキーのデフォルトを追加する必要があります**。Certificate System は、サブジェクトキー識別子のデフォルトを作成または適用する前にキーのデフォルトで鍵制約を処理するため、鍵がまだ処理されていない場合は、サブジェクト名に鍵の設定に失敗します。

たとえば、object-signing プロファイルでは両方のデフォルト値を定義できます。

```
policyset.set1.p3.constraint.class_id=noConstraintImpl
policyset.set1.p3.constraint.name=No Constraint
policyset.set1.p3.default.class_id=subjectKeyIdentifierExtDefaultImpl
policyset.set1.p3.default.name=Subject Key Identifier Default
...
policyset.set1.p11.constraint.class_id=keyConstraintImpl
policyset.set1.p11.constraint.name=Key Constraint
policyset.set1.p11.constraint.params.keyType=RSA
policyset.set1.p11.constraint.params.keyParameters=1024,2048,3072,4096
policyset.set1.p11.default.class_id=userKeyDefaultImpl
policyset.set1.p11.default.name=Key Default
```

policyset リストでは、サブジェクトキー識別子のデフォルト (**p3**) の前にキーのデフォルト (**p11**) を指定する必要があります。

```
policyset.set1.list=p1,p2,p11,p3,p4,p5,p6,p7,p8,p9,p10
```

3.4. 更新を有効にするためのプロファイルの設定

本セクションでは、証明書更新にプロファイルを設定する方法を説明します。証明書の更新方法は、「証明書の更新」 を参照してください。

更新を許可するプロファイルは、**updateGracePeriodConstraint** エントリーで頻繁に発生します。以下に例を示します。

```
policyset.cmcUserCertSet.10.constraint.class_id=renewGracePeriodConstraintImpl
policyset.cmcUserCertSet.10.constraint.name=Renewal Grace Period Constraint
policyset.cmcUserCertSet.10.constraint.params.renewal.graceBefore=30
policyset.cmcUserCertSet.10.constraint.params.renewal.graceAfter=30
policyset.cmcUserCertSet.10.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.10.default.name=No Default
```

3.4.1. 同じ鍵を使用した更新

更新に同じキーの送信を可能にするプロファイルで、**uniqueKeyConstraint** エントリーの ***allowSameKeyRenewal*** パラメーターが **true** に設定されています。以下に例を示します。

```
policyset.cmcUserCertSet.9.constraint.class_id=uniqueKeyConstraintImpl
policyset.cmcUserCertSet.9.constraint.name=Unique Key Constraint
policyset.cmcUserCertSet.9.constraint.params.allowSameKeyRenewal=true
policyset.cmcUserCertSet.9.default.class_id=noDefaultImpl
policyset.cmcUserCertSet.9.default.name=No Default
```

3.4.2. 新しい鍵を使用した更新

新しい鍵で証明書を更新するには、新しいキーで同じプロファイルを使用します。Certificate System は、新しい証明書の要求の署名に使用されるユーザー署名証明書の **subjectDN** を使用します。

3.5. 証明書の署名アルゴリズムの設定

CA の署名証明書は、CA でサポートされる公開鍵アルゴリズムに問題がある証明書に署名できます。たとえば、ECC 署名証明書は、ECC アルゴリズムと RSA アルゴリズムの両方が CA でサポートされている限り、ECC 証明書要求と RSA 証明書要求の両方に署名できます。RSA 署名証明書は EC キーを使用して PKCS # 10 要求に署名できますが、CA が CRMF 所有証明 (POP) を検証するために ECC モジュールを使用できない場合は、EC キーを使用して CRMF 証明書要求に署名できない場合があります。

ECC および RSA は、公開鍵の暗号化と署名アルゴリズムです。両方の公開鍵アルゴリズムは、異なる暗号スイートをサポートします。これは、データの暗号化と復号に使用されるアルゴリズムです。CA 署名証明書の機能には、対応している暗号スイートのいずれかを使用して、証明書を発行し、署名することです。

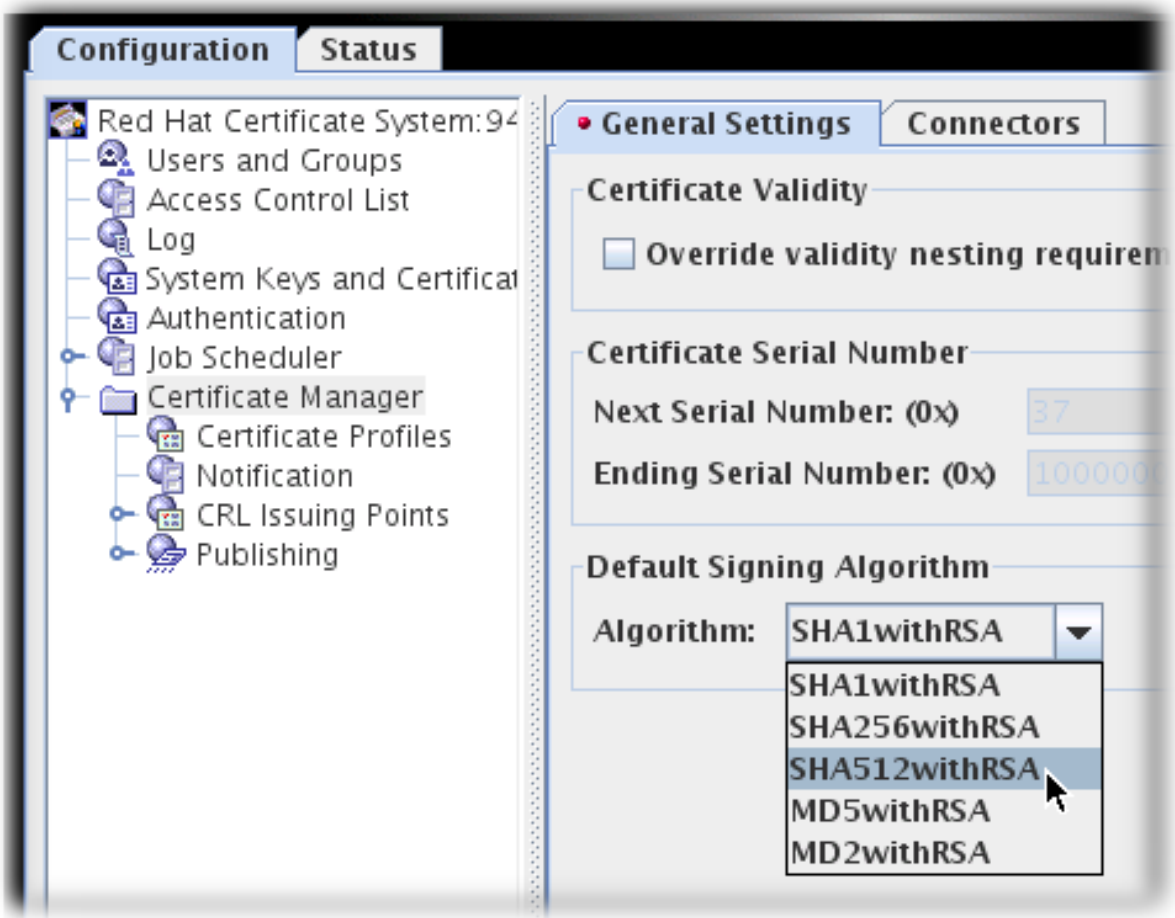
各プロファイルは、CA がそのプロファイルで処理される証明書の署名に使用する暗号化スイートを定義できます。署名アルゴリズムが設定されていない場合、プロファイルはデフォルトの署名アルゴリズムを使用します。

3.5.1. CA のデフォルト署名アルゴリズムの設定

1. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、**Certificate Manager** ツリーを展開します。
3. **General Settings** タブで、**Algorithm** ドロップダウンメニューで使用するアルゴリズムを設定します。



注記

pkiconsole が非推奨になりました。

3.5.2. プロファイルでの署名アルゴリズムのデフォルトの設定

各プロファイルには、Signing Algorithm Default 拡張が定義されています。デフォルトには、デフォルトのアルゴリズムと、証明書要求で別のアルゴリズムが指定されている場合に許可されるアルゴリズムのリストの 2 つの設定があります。署名アルゴリズムが指定されていない場合、プロファイルは CA のデフォルトとして設定されているものを使用します。

プロファイルの **.cfg** ファイルで、アルゴリズムは 2 つのパラメーターで設定されます。

```
policyset.cmcUserCertSet.8.constraint.class_id=signingAlgConstraintImpl
policyset.cmcUserCertSet.8.constraint.name=No Constraint
policyset.cmcUserCertSet.8.constraint.params.signingAlgsAllowed=SHA256withRSA,SHA512withRSA,S
HA256withEC,SHA384withRSA,SHA384withEC,SHA512withEC
policyset.cmcUserCertSet.8.default.class_id=signingAlgDefaultImpl
policyset.cmcUserCertSet.8.default.name=Signing Alg
policyset.cmcUserCertSet.8.default.params.signingAlg=-
```

コンソールから Signing Algorithm Default を設定するには、以下を行います。

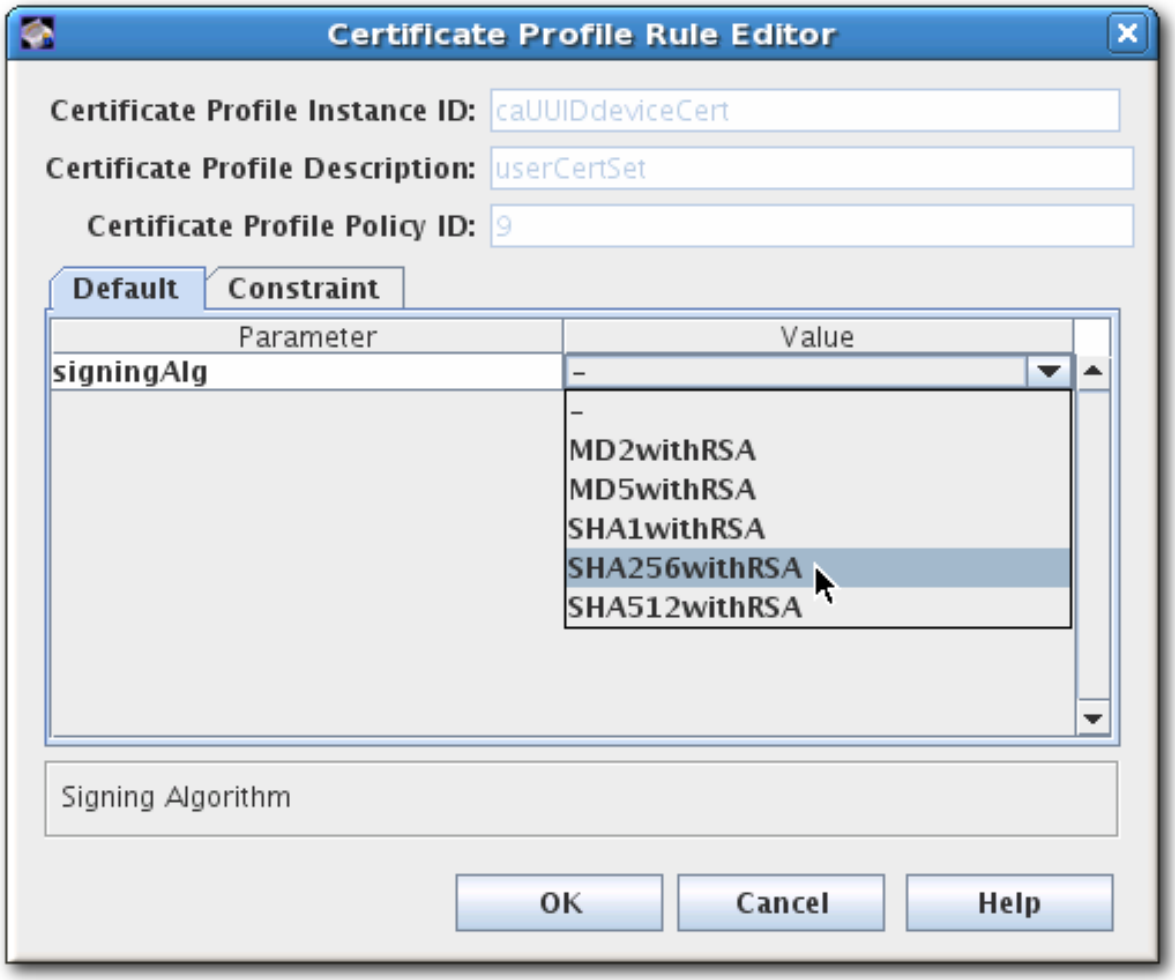
注記

プロファイルを編集する前に、エージェントにより最初に無効にする必要があります。

1. CA コンソールを開きます。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで、**Certificate Manager** ツリーを展開します。
3. **Certificate Profiles** 項目をクリックします。
4. **Policies** タブをクリックします。
5. **Signing Alg ポリシー** を選択して **Edit** ボタンをクリックします。
6. デフォルトの署名アルゴリズムを設定するには、**Defaults** タブで値を設定します。これが **-** に設定されている場合、プロファイルは CA のデフォルトを使用します。



7. 証明書要求で許可される署名アルゴリズムの一覧を設定するには、**Constraints** タブを開き、**singiningAlgsAllowed** の **Value** フィールドでアルゴリズムの一覧を設定します。

   制約に使用できる値は、「アルゴリズム制約の署名」 に記載されています。

注記

pkiconsole が非推奨になりました。

3.6. CA 関連プロファイルの管理

証明書プロファイルと拡張機能を使用して、下位 CA が証明書を発行する方法にルールを設定する必要があります。これには 2 つの部分があります。

- CA 署名証明書の管理
- 発行ルールの定義

3.6.1. CA 証明書での制限の設定

下位 CA が作成されると、ルート CA は下位 CA に制限または制限を課できます。たとえば、ルート CA は、CA 署名証明書の Basic Constraints 拡張機能の pathLenConstraint フィールドを設定することにより、有効な認証パスの最大深度 (新しい CA の下にチェーンできる下位 CA の数) を指定できます。

証明書チェーンは、通常エンティティー証明書、ゼロまたは中間 CA 証明書、ルート CA 証明書で設定されます。ルート CA 証明書は、自己署名型または外部の信頼できる CA によって署名されます。ルート CA 証明書は、信頼できる CA として証明書データベースに読み込まれます。

証明書の交換は、TLS ハンドシェイクの実行時、S/MIME メッセージの送信時、または署名済みオブジェクトを送信するときに行われます。ハンドシェイクの一部として、送信者は、サブジェクト証明書と、サブジェクト証明書を信頼されたルートにリンクするために必要な中間 CA 証明書を送信する必要があります。証明書チェーンが適切に証明書を有効にするには、以下のプロパティーが必要です。

- CA 証明書には、基本的な制約の拡張子が必要です。
- CA 証明書の鍵用途拡張に keyCertSign ビットが設定されている必要があります。
- CA が新しい鍵を生成する場合は、すべてのサブジェクト証明書に認証局キー識別子の拡張子を追加する必要があります。この拡張機能は、これらの証明書を古い CA 証明書と区別するのに役立ちます。CA 証明書には Subject Key Identifier 拡張が含まれている必要があります。

証明書とその拡張の詳細は、RFC 5280 で利用可能な 『Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile (RFC 5280)』 を参照してください。

これらの拡張機能は、証明書プロファイルの登録ページで設定できます。デフォルトでは、CA には必須かつ合理的な設定設定が含まれますが、これらの設定をカスタマイズすることは可能です。

注記

この手順では、CA が使用する CA 証明書プロファイルを編集して、下位 CA に CA 証明書を発行する方法を説明します。

CA インスタンスの初期設定時に使用されるプロファイルは、**/var/lib/pki/*instance_name*/ca/conf/caCert.profile** です。このプロファイルは、**pkiconsole** で編集することはできません (インスタンスを設定する前のみ利用可能)。テキストエディターで CA を設定する前に、テンプレートファイルでこのプロファイルのポリシーを編集することができます。

CA が使用する CA 署名証明書プロファイルでデフォルトを変更するには、以下を実行します。

1. プロファイルが有効になっている場合は、編集する前に無効にする必要があります。エージェントサービスページを開き、左側のナビゲーションメニューから **Manage Certificate Profile** を選択してプロファイルを選択し、**Disable profile** をクリックします。
2. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

3. **Configuration** タブの左側のナビゲーションツリーで、**Certificate Manager** を選択し、**Certificate Profiles** を選択します。
4. 右側のウィンドウから caCACert または該当する CA 署名証明書プロファイルを選択して、**Edit/View** をクリックします。
5. **Certificate Profile Rule Editor** の **Policies** タブで、キー使用法または拡張キー使用法拡張機能のデフォルトが存在する場合はそれを選択して編集するか、プロファイルに追加します。
6. 必要に応じて、デフォルトとして、Key Usage または Extended Key Usage Extension Constraint を選択します。

7. CA 証明書のデフォルト値を設定します。詳細は、「Key Usage 拡張機能のデフォルト」および「Extended Key Usage 拡張機能のデフォルト」を参照してください。

8. CA 証明書の制約値を設定します。キー使用拡張に設定する制約はありません。拡張キーの使用の拡張機能の場合は、CA に適切な OID 制約を設定します。詳細は、「Extended Key Usage 拡張機能のデフォルト」 を参照してください。

9. プロファイルに変更を加えたら、エージェントサービスページを再度ログインして、証明書プロファイルを再度有効にします。

注記

pkiconsole が非推奨になりました。

証明書プロファイルの変更の詳細は、「証明書プロファイルの設定」 を参照してください。

3.6.2. 証明書の発行における CA の制限の変更

発行された証明書の制限は、サブシステムの設定後にデフォルトで設定されます。これには、以下が含まれます。

- CA 署名証明書よりも長い有効期間で証明書を発行できるかどうか。デフォルトでは、これを無効にします。
- 証明書の署名に使用される署名アルゴリズム。
- CA が証明書を発行するために使用するシリアル番号の範囲。

下位 CA には、有効期間、証明書の種類、および発行可能な拡張の種類に制約があります。下位 CA はこれらの制約に違反する証明書を発行できますが、これらの制約に違反する証明書を認証するクライアントはその証明書を受け入れません。下位 CA の発行ルールを変更する前に、CA 署名証明書に設定された制約を確認してください。

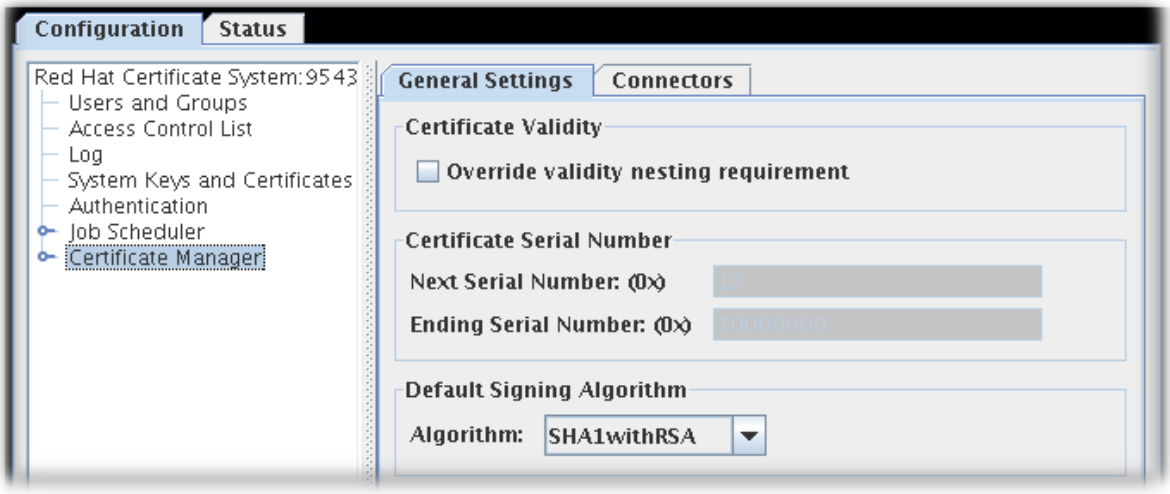
証明書の発行ルールを変更するには、次のコマンドを実行します。

1. 証明書システムコンソールの起動

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブの左側のナビゲーションツリーで、**Certificate Manager** アイテムを選択します。

図3.1デフォルトでは、非従属 CA の一般設定タブ



3. デフォルトでは、クローン以外の CA では、**Certificate Manager** メニュー項目の **General Settings** タブに以下のオプションが含まれます。

   - **Override validity nesting requirement**。このチェックボックスでは、Certificate Manager が、CA 署名の証明書有効期間よりも長い有効期間の証明書を発行できるかどうかを設定します。

     このチェックボックスを選択しておらず、CA が CA 署名証明書の有効期間よりも長い期間要求を受け取ると、CA 署名証明書の期限が切れる時点で自動的に終了するように有効期間が切り捨てられます。

   - **Certificate Serial Number**。これらのフィールドは、Certificate Manager が発行する証明書のシリアル番号の範囲を表示します。サーバーは、**Next serial number** を、次に発行する証明書に割り当て、**Ending serial number** を、最後に発行した証明書に割り当てます。

     シリアル番号の範囲により、複数の CA をデプロイでき、各 CA が発行する証明書の数のバランスを取ります。発行者名とシリアル番号の組み合わせは、証明書を一意に識別する必要があります。

注記

クローン CA を使用するシリアル番号の範囲は fluid です。複製されたすべての CA は、次の利用可能な範囲を定義する共通の設定エントリーを共有します。1 つの CA が利用可能な数未満の実行を開始すると、この設定エントリーをチェックし、次の範囲を要求します。エントリーは自動的に更新されます。これにより、次の CA が新規範囲を取得します。

範囲は ***begin*Number*** 属性および ***end*Number*** 属性で定義され、個別の範囲が要求および証明書のシリアル番号に対して定義されます。以下に例を示します。

```
dbs.beginRequestNumber=1
dbs.beginSerialNumber=1
dbs.enableSerialManagement=true
dbs.endRequestNumber=9980000
dbs.endSerialNumber=ffe0000
dbs.ldap=internaldb
dbs.newSchemaEntryAdded=true
dbs.replicaCloneTransferNumber=5
```

シリアル番号管理は、クローンされていない CA に対して有効にできます。ただし、デフォルトでは、システムが自動的に有効になった場合にシステムのクローンが作成されない限り、シリアル番号の管理が無効になります。

シリアル番号の範囲は、コンソールで手動で更新することはできません。シリアル番号の範囲は読み取り専用フィールドです。

- **署名アルゴリズムのデフォルト。** Certificate Manager が証明書の署名に使用する署名アルゴリズムを指定します。このオプションは、CA の署名鍵タイプが RSA の場合は、**SHA256withRSA** および **SHA512withRSA** です。

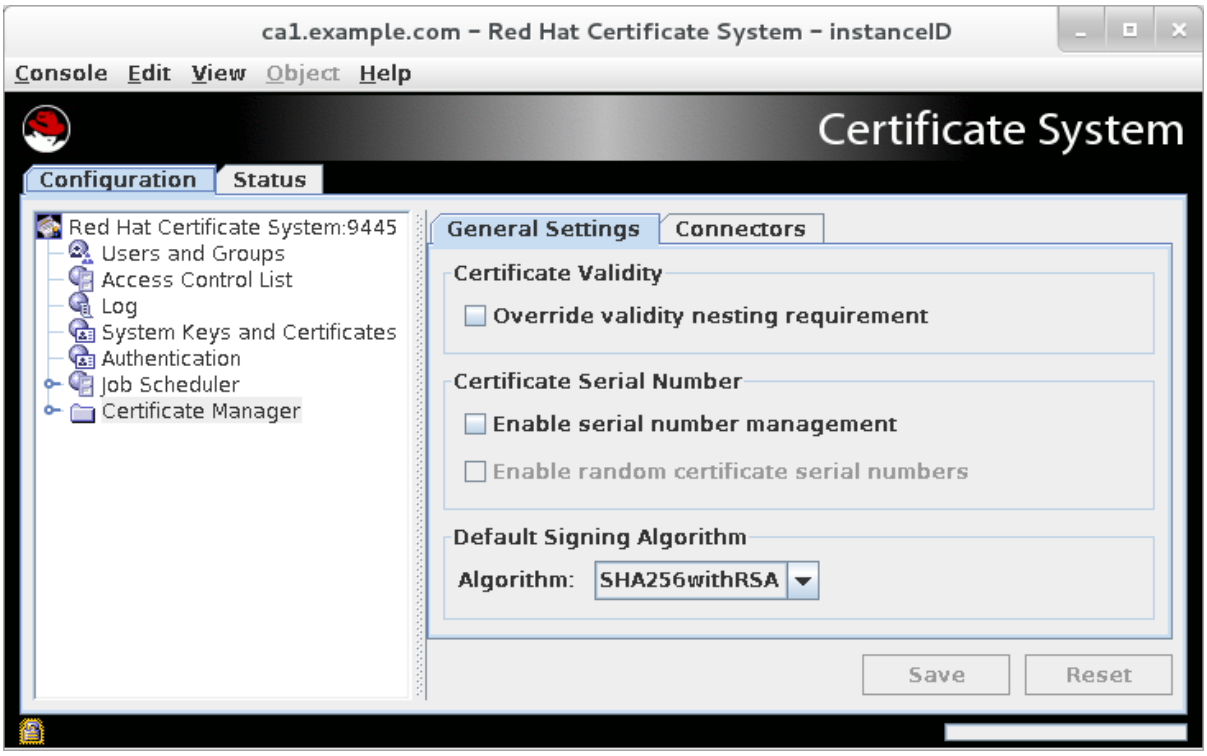
  証明書プロファイル設定に指定された署名アルゴリズムは、ここに設定されたアルゴリズムよりも優先されます。

4. デフォルトでは、クローン作成された CA では、**Certificate Manager** メニュー項目の **General Settings** タブに以下のオプションが含まれます。

   - **ランダムなシリアル番号管理**
   - **Enable random certificate serial numbers**

   両方のチェックボックスを選択します。

図3.2 デフォルトでクローン作成された CA の General Settings タブ



5. **Save** をクリックします。

注記

pkiconsole が非推奨になりました。

3.6.3. ランダム証明書のシリアル番号の使用

Red Hat Certificate System には、要求、証明書、レプリカ ID に対するシリアル番号の範囲管理が含まれています。これにより、**Identity Management** (IdM) インストール時のクローン作成を自動化できます。

ハッシュベースの攻撃の可能性が低くなるには、以下の方法を使用できます。

- 攻撃者に予測できない証明書のシリアル番号の一部にする
- ランダムに選択されたコンポーネントを ID に追加する
- それぞれを前後に歪めることにより、攻撃者が有効期限を予測できないようにする

ランダムな証明書のシリアル番号割り当て 方法は、無作為に選択されたコンポーネントを ID に追加します。この方法は以下の通りです。

- クローン作成で機能
- 競合の解決を許可する
- 現在のシリアル番号管理方法との互換性がある
- 管理者、エージェント、およびエンドエンティティーの現在のワークフローと互換性がある
- 連続するシリアル番号管理で既存のバグを修正。

備考

管理者は、証明書のシリアル番号を有効にする必要があります。

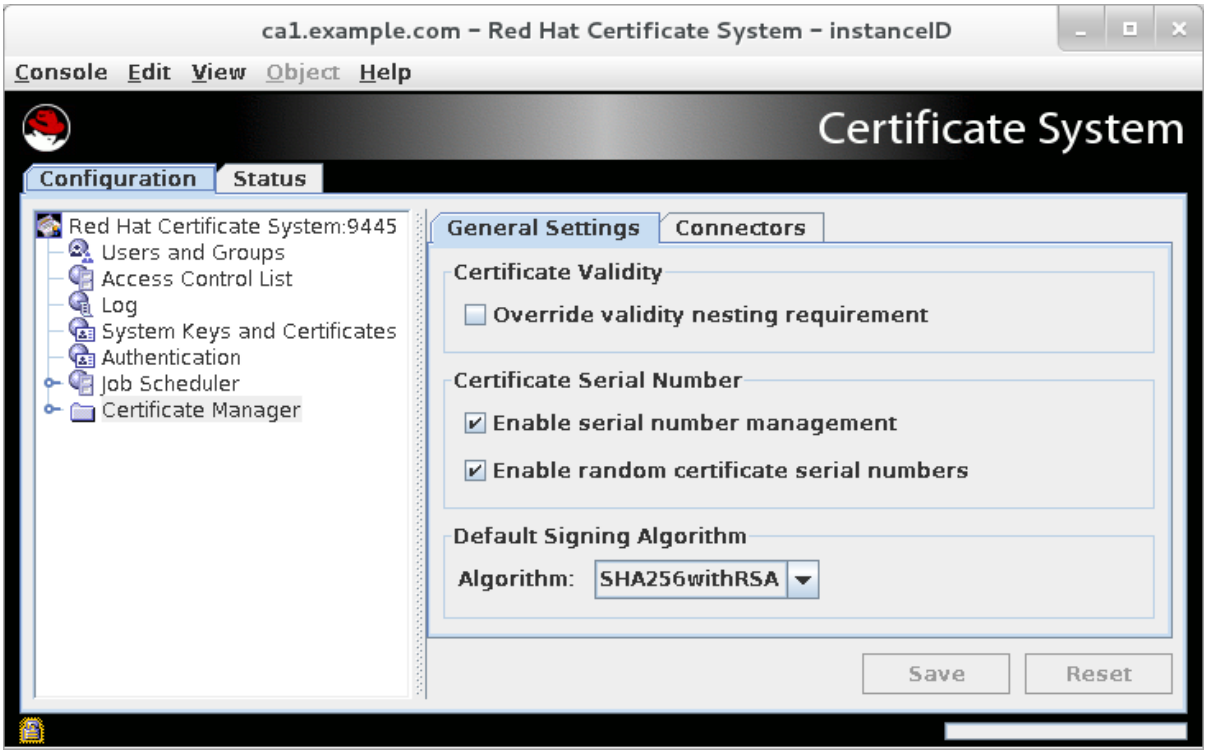
3.6.3.1. ランダム証明書のシリアル番号の有効化

コマンドラインまたはコンソール UI から自動シリアル番号範囲管理を有効にすることができます。

コンソール UI から自動シリアル番号管理を有効にするには、以下を行います。

1. **General Settings** タブで、**Enable serial number management** オプションを選択します。

図3.3 乱数の割り当てが有効な場合の General Settings タブ



2. **Enable random certificate serial numbers** オプションをオンにします。

注記

pkiconsole が非推奨になりました。

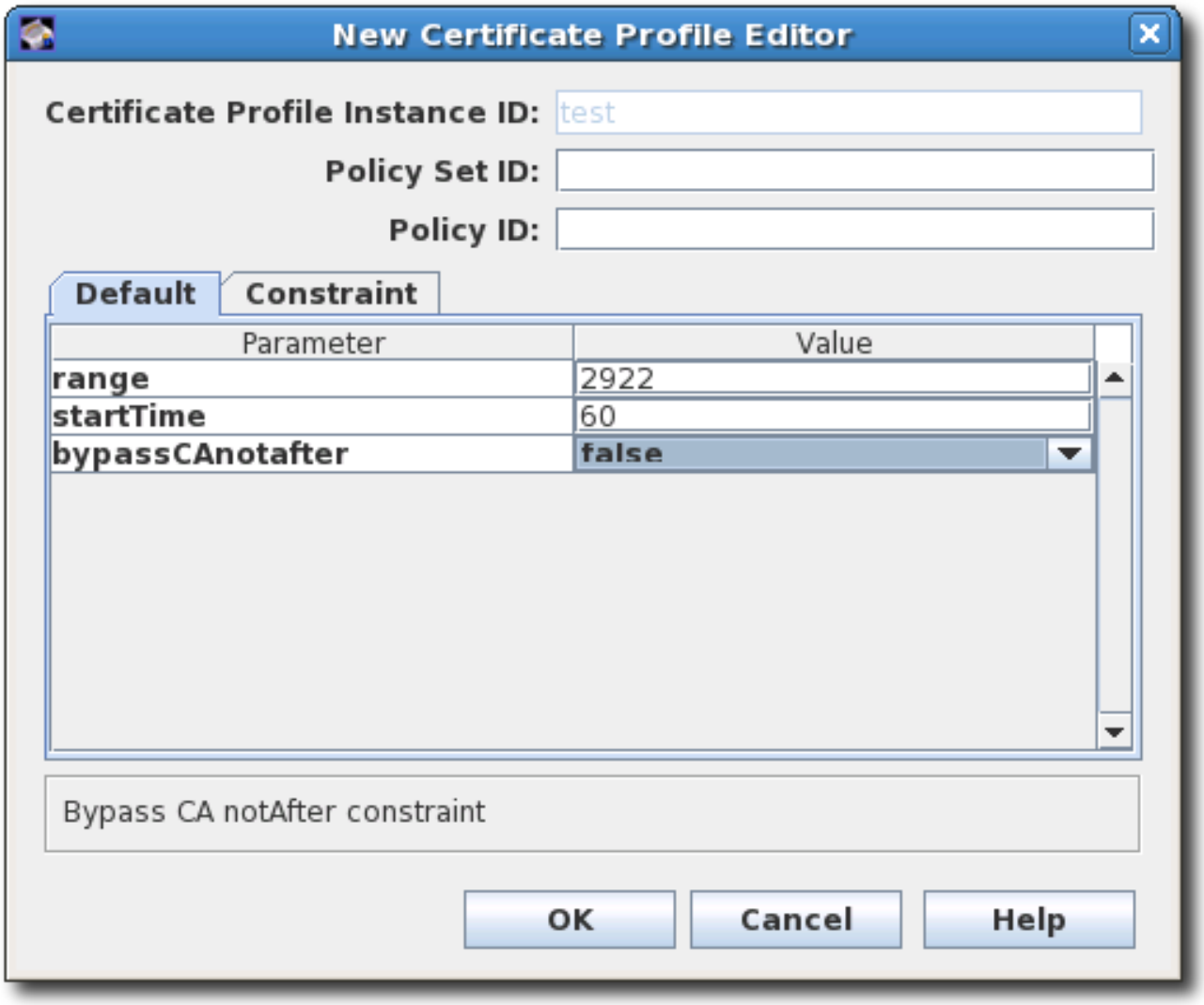
3.6.4. 認証局の有効期間を過ぎた認証局証明書の更新の許可

通常、証明書は、発行先の CA 証明書の有効期限 **後** に終わる有効期間では発行できません。CA 証明書の有効期限が 2015 年 12 月 31 日である場合、証明書はすべて 2015 年 12 月 31 日または 2015 年 12 月 31 日より前に有効期限が切れる必要があります。

このルールは、CA が発行する他の CA 署名証明書に適用されます。これにより、ルート CA 証明書はほとんど更新できなくなります。CA 署名証明書を更新するということは、それ自体の有効期限を過ぎた有効期間が必ず必要になることを意味します。

この動作は CA Validity Default を使用して変更できます。このデフォルト設定では、発行している CA の有効期限 (**notAfter**) の期間を拡張する有効期間で CA 証明書を発行できる設定 (**bypassCAnotafter**) が許可されます。

図3.4 CA 有効性のデフォルト設定



実際のデプロイメントでは、これが意味するのは、ルート CA の CA 証明書は、他の方法では防止できる場合でも更新できるということです。

元の CA の有効期間前に CA 証明書の更新を有効にするには、以下を実行します。

1. **caCACert.cfg** ファイルを開きます。

```
vim /var/lib/pki/instance_name/ca/conf/caCACert.cfg
```

2. CA Validity デフォルトはデフォルトで存在する必要があります。値を **true** に設定して、発行 CA の有効期間を過ぎて CA 証明書を更新できるようにします。

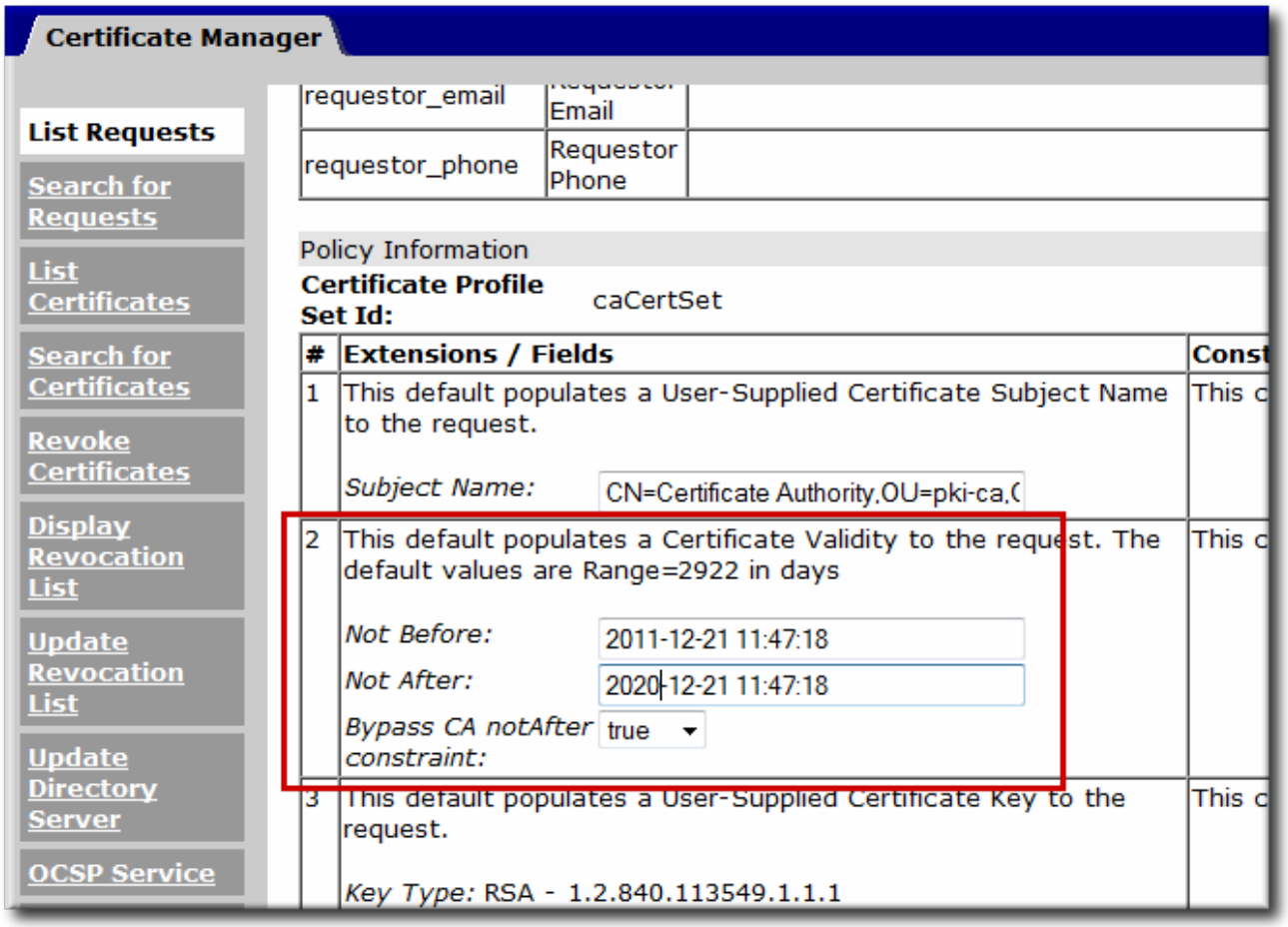
```
policyset.caCertSet.2.default.name=CA Certificate Validity Default
policyset.caCertSet.2.default.params.range=2922
policyset.caCertSet.2.default.params.startTime=0
policyset.caCertSet.2.default.params.bypassCAnotafter=true
```

3. CA を再起動して変更を適用します。

エージェントが更新要求を確認すると、通常の有効期間制約をバイパスすることを可能にする **Extensions/Fields** エリアにオプションがあります。エージェントが **false** を選択すると、プロファイルで **bypassCAnotafter=true** が設定されていても制約が適用されます。**bypassCAnotafter** 値が有効

になっていない時にエージェントが true を選択すると、更新要求は CA によって拒否されます。

図3.5 エージェントサービスページの CA 制約オプションを回避



注記

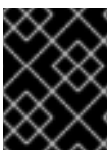
CA Validity のデフォルトは、CA 署名の証明書の更新にのみ適用されます。その他の証明書は、引き続き CA の有効期間内で発行および更新する必要があります。

CA の **ca.enablePastCATime** の個別の設定を使用すると、CA の有効期間を過去に証明書を更新することができます。ただし、これは、その CA が発行する **すべて** の証明書に適用されます。セキュリティーの問題が発生する可能性があるため、実稼働環境でこの設定は推奨されません。

3.7. サブジェクト名およびサブジェクト代替名の管理

証明書の **サブジェクト名** は、証明書を発行するエンティティーの ID 情報が含まれる識別名 (DN) です。このサブジェクト名は、共通名や組織単位などの標準の LDAP ディレクトリーコンポーネントから構築できます。これらのコンポーネントは X.500 に定義されます。サブジェクト名の他に、証明書には **サブジェクト代替名** があります。これは、X.500 に定義されていない追加情報が含まれる証明書の拡張機能セットです。

サブジェクト名とサブジェクトの代替名の命名コンポーネントはカスタマイズできます。



重要

サブジェクト名が空の場合は、Subject Alternative Name 拡張が存在し、critical のマークが付けられている必要があります。

3.7.1. サブジェクト名でのリクエスター CN または UID の使用

証明書要求の ***cn*** 値または ***uid*** 値は、発行した証明書のサブジェクト名をビルドするために使用できます。このセクションでは、サブジェクト名制約に naming 属性 (CN または UID) が証明書要求に存在する必要があるプロファイルを示しています。naming 属性がないと、リクエストは拒否されます。

この設定には、以下の 2 つの部分があります。

- CN または UID 形式は、Subject Name Constraint の **pattern** 設定に設定されます。
- CN または UID トークン、および証明書の特定の接尾辞を含むサブジェクト DN の形式は、Subject Name Default に設定されます。

たとえば、サブジェクト DN で CN を使用するには、次のコマンドを実行します。

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=CN=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=CN=$request.req_subject_name.cn$,DC=example,
DC=com
```

この例では、リクエストに **cn=John Smith** の CN が含まれる場合、証明書は、 **cn=John Smith,DC=example, DC=com** のサブジェクト DN で発行されます。要求が完了しても、 **uid=jsmith** の UID がありCN がない場合、要求は拒否されます。

同じ設定を使用して、要求側の UID をサブジェクト DN にプルします。

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=UID=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=UID=$request.req_subject_name.uid$,DC=example,
DC=com
```

pattern パラメーターの形式は、 「Subject Name 制約」 および 「サブジェクト名のデフォルト」 で説明されています。

3.7.2. サブジェクト代替名への LDAP ディレクトリー属性値およびその他の情報の挿入

LDAP ディレクトリーからの情報、または要求元によって送信された情報は、Subject Alt Name Extension Default 設定で一致する変数を使用して、証明書のサブジェクト代替名に挿入できます。デフォルトでは、情報のタイプ (形式) と、情報の取得に使用する一致するパターン (変数) を設定します。以下に例を示します。

```
policyset.userCertSet.8.default.class_id=subjectAltNameExtDefaultImpl
policyset.userCertSet.8.default.name=Subject Alt Name Constraint
policyset.userCertSet.8.default.params.subjAltNameExtCritical=false
policyset.userCertSet.8.default.params.subjAltExtType_0=RFC822Name
policyset.userCertSet.8.default.params.subjAltExtPattern_0=$request.requestor_email$
policyset.userCertSet.8.default.params.subjAltExtGNEnable_0=true
```

これにより、要求側の電子メールがサブジェクト名の最初の CN コンポーネントとして挿入されます。追加のコンポーネントを使用するには、**Type_**、**Pattern_**、および **Enable_** 値を、**Type_1** などの数値にインクリメントします。

サブジェクトの alt 名の設定については、「サブジェクト代替名の拡張機能のデフォルト」 を参照してください。

LDAP コンポーネントを証明書のサブジェクト代替名に挿入するには、以下を実行します。

1. LDAP 属性値を挿入するには、ユーザーディレクトリーの認証プラグイン **SharedSecret** を有効にする必要があります。

   a. CA コンソールを開きます。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

   b. 左側のナビゲーションツリーで **認証** を選択します。

   c. **Authentication Instance** タブで、**Add** をクリックして、**SharedSecret** 認証プラグインのインスタンスを追加します。

   d. 以下の情報を入力します。

   ```
   Authentication InstanceID=SharedToken
   shrTokAttr=shrTok
   ldap.ldapconn.host=server.example.com
   ldap.ldapconn.port=636
   ldap.ldapconn.secureConn=true
   ldap.ldapauth.bindDN=cn=Directory Manager
   password=password
   ldap.ldapauth.authtype=BasicAuth
   ldap.basedn=ou=People,dc=example,dc=org
   ```

   e. 新規プラグインインスタンスを保存します。

   注記

   pkiconsole が非推奨になりました。

   JOIN 共有トークンの設定に関する詳細は、「CMC 共有シークレットの設定」 を参照してください。

2. **ldapStringAttributes** パラメーターは、ユーザーの LDAP エントリーから ***mail*** 属性の値を読み込み、その値を証明書要求に追加するように、認証プラグインに指示します。値が要求に設定されている場合、証明書プロファイルポリシーは、拡張値のその値を挿入するように設定できます。

   dnpattern パラメーターの形式は、「Subject Name 制約」 および 「サブジェクト名のデフォルト」 で説明されています。

3. CA が証明書拡張機能に LDAP 属性の値を挿入できるようにするには、プロファイルの設定ファイルを編集し、拡張機能のポリシーセットパラメーターを挿入します。たとえば、**caFullCMCSharedTokenCert** プロファイルの Subject Alternative Name 拡張に ***mail*** 属性値を挿入するには、以下のコードを変更します。

```
policyset.setID.8.default.params.subjAltExtPattern_0=$request.auth_token.mail[0]$
```

プロファイルの編集に関する詳細は、「RAW 形式での証明書プロファイルの編集」 を参照してください。

4. CA を再起動します。

```
systemctl restart pki-tomcatd-nuxwdog@instance_name.service
```

この例では、**caFullCMCSharedTokenCert** プロファイル登録フォームを介して送信される証明書では、要求側の ***mail*** LDAP 属性の値で Subject Alternative Name 拡張機能が追加されます。以下に例を示します。

```
Identifier: Subject Alternative Name - 2.5.29.17
    Critical: no
    Value:
    RFC822Name: jsmith@example.com
```

このポリシーセットのいずれかの **Pattern_** パラメーターにトークン (**X**) として設定することにより、証明書に自動的に挿入できる属性が多数あります。一般的なトークンを 表3.1「証明書の設定に使用する変数」 に記載します。デフォルトのプロファイルに、これらのトークンの使用方法の例が含まれています。

表3.1 証明書の設定に使用する変数

ポリシーセットトークン	説明
$request.auth_token.cn[0]$	証明書を要求したユーザーの LDAP 共通名 (***cn***) 属性。
$request.auth_token.mail[0]$	証明書を更新したユーザーの LDAP メール (***mail***) 属性の値。
$request.auth_token.tokencertsubject$	証明書サブジェクト名。
$request.auth_token.uid$	証明書を要求したユーザーの LDAP ユーザー ID (***uid***) 属性。
$request.auth_token.userdn$	証明書を要求したユーザーのユーザー DN。
$request.auth_token.userid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.uid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.requestor_email$	要求を送信したユーザーのメールアドレス。
$request.requestor_name$	要求を送信した人。

ポリシーセットトークン	説明
$request.upn$	Microsoft UPN。これには **(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.upn$** の形式があります。
$server.source$	サーバーに対し、サブジェクト名のバージョン 4 の UUID (乱数) コンポーネントを生成するように指示します。この値は常に **(IA5String)1.2.3.4,$server.source$** 形式になります。
$request.auth_token.user$	要求が TPS によって送信された場合に使用します。証明書をリクエストした TPS サブシステム信頼マネージャー。
$request.subject$	要求が TPS によって送信された場合に使用します。TP S が解決して要求したエンティティーのサブジェクト名 DN。例: **cn=John.Smith.123456789,o=TMS Org**

3.7.3. SAN 拡張での CN 属性の使用

RFC 2818 で非推奨となったドメイン名の検証に、サブジェクト DN の Common Name (CN) 属性の使用に対応しなくなりました。代わりに、これらのアプリケーションやライブラリーは、証明書要求で ***dNSName*** Subject Alternative Name (SAN) の値を使用します。

Certificate System は、RFC 1034 セクション 3.5 に従って優先名前構文と一致し、複数のコンポーネントを持つ場合にのみ CN をコピーします。また、既存の SAN 値が保持されます。たとえば、CN をベースとする ***dNSName*** 値は、既存の SAN に追加されます。

SAN 拡張の CN 属性を使用するように Certificate System を設定するには、証明書を発行するために使用する証明書プロファイルを編集します。以下に例を示します。

1. プロファイルを無効にします。

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-disable profile_name
```

2. プロファイルを編集します。

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-edit profile_name
```

   a. プロファイルに固有のセット番号を使用して、以下の設定を追加します。以下に例を示します。

```
policyset.serverCertSet.12.constraint.class_id=noConstraintImpl
policyset.serverCertSet.12.constraint.name=No Constraint
policyset.serverCertSet.12.default.class_id=commonNameToSANDefaultImpl
policyset.serverCertSet.12.default.name=Copy Common Name to Subject
```

前述の例では、**12** をセット番号として使用しています。

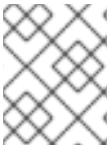
b. ***policyset.userCertSet.list*** パラメーターに新しいポリシーセット番号を追加します。以下に例を示します。

```
policyset.userCertSet.list=1,10,2,3,4,5,6,7,8,9,12
```

c. プロファイルを保存します。

3. プロファイルを有効にします。

```
# pki -c password -p 8080 \
    -n "PKI Administrator for example.com" ca-profile-enable profile_name
```



注記

すべてのデフォルトサーバーグループに、***commonNameToSANDefaultImpl*** のデフォルトが含まれます。

3.7.4. CSR からの SAN 拡張の許可

特定の環境では、管理者は Certificate Signing Request (CSR) で Subject Alternative Name (SAN) 拡張機能を指定できるようにします。

3.7.4.1. CSR から SAN を取得するプロファイルの設定

CSR からの SAN の取得を許可するには、ユーザー拡張機能のデフォルトを使用します。詳細は、「User Supplied Extension Default」 を参照してください。

注記

SAN 拡張には、1 つ以上の SAN を含めることができます。

CSR から SAN を受け入れるには、**caCMCECserverCert** のように、以下のデフォルトおよび制約をプロファイルに追加します。

```
prefix.constraint.class_id=noConstraintImpl
prefix.constraint.name=No Constraint

prefix.default.class_id=userExtensionDefaultImpl
prefix.default.name=User supplied extension in CSR
prefix.default.params.userExtOID=2.5.29.17
```

3.7.4.2. SAN を使用した CSR の生成

たとえば、**certutil** ユーティリティーを使用して、2 つの SAN を持つ CSR を生成するには、以下を実行します。

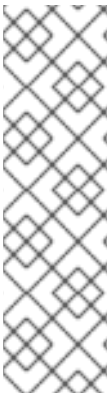
```
# certutil -R -k ec -q nistp256 -d . -s "cn=Example Multiple SANs" --extSAN
dns:www.example.com,dns:www.example.org -a -o /root/request.csr.p10
```

CSR の生成後に、「CMC 登録プロセス」 で説明されている手順に従って、CMC の登録を完了します。

第4章 キーアーカイブおよびリカバリーの設定

キーアーキタイプおよびリカバリーの詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『キーのアーカイブ、リカバリー、およびローテーション』 セクションを参照してください。

この章では、以前は Data Recovery Manager (DRM) と呼ばれていた Key Recovery Authority (KRA) を設定して、秘密鍵をアーカイブし、暗号化されたデータを復元するためにアーカイブされた鍵を回復する方法を説明します。

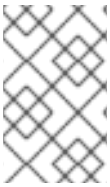


注記

本章では、クライアント側の鍵生成を使用した鍵のアーカイブのみを説明します。サーバー側のキーの生成とアーカイブは、TPS を介して開始されたか、CA のエンドエンティティーポータルを介して開始されたかにかかわらず、ここでは説明しません。

スマートカード鍵のリカバリーの詳細は、「サーバー側の鍵生成の設定」 を参照してください。

CA の EE ポータルで提供されるサーバー側の鍵生成に関する詳細は、「サーバー側の鍵生成を使用した CSR の生成」 を参照してください。



注記

Gemalto SafeNet LunaSA は、CKE - Key Export モデルでの PKI 秘密鍵抽出のみおよび非 FIPS モードでのみサポートされます。FIPS モードの LunaSA Cloning モデルおよび CKE モデルは、PKI 秘密鍵の抽出をサポートしません。

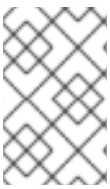
KRA がインストールされると、セキュリティードメインに参加し、CA とペアになります。このような場合、秘密暗号化キーをアーカイブおよび復元するように設定されています。ただし、KRA 証明書がセキュリティードメイン内の CA のいずれかではなく外部 CA によって発行される場合は、キーのアーカイブとリカバリープロセスを手動で設定する必要があります。

詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『キーアーカイブの手動設定』 セクションを参照してください。

注記

クローン環境では、キーのアーカイブとリカバリーを手動で設定する必要があります。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『クローン後の CA-KRA コネクター情報の更新』セクションを参照してください。

4.1. コンソールでのエージェント承認キーリカバリーの設定



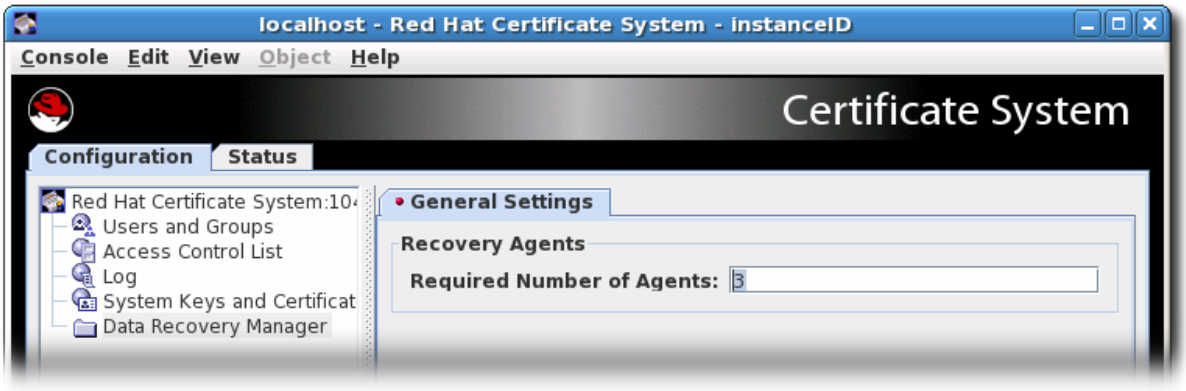
注記

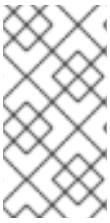
コンソールでキーリカバリーエージェントの **数** を設定できますが、使用する **グループ** は **CS.cfg** ファイルで直接設定できます。コンソールはデフォルトで **Key Recovery Authority Agents Group** を使用します。

1. KRA のコンソールを開きます。以下に例を示します。

```
pkiconsole https://server.example.com:8443/kra
```

2. 左側のナビゲーションツリーの **Key Recovery Authority** のリンクをクリックします。
3. **Required Number of Agents** フィールドに、キー復元を承認するのに使用するエージェントの数を入力します。





注記

エージェントが承認したキー復元を **CS.cfg** ファイルで設定する詳細な方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の 『コマンドラインでのエージェント承認キーリカバリーの設定』 を参照してください。

4.2. キーアーカイブおよびリカバリー設定のテスト

注記

新しいブラウザーは、ブラウザーからのキーのアーカイブをサポートしていません。ステップ 1 では、これらのブラウザーの代わりに、『CRMF 生成クライアント』を使用する必要があります。

鍵を正常にアーカイブできるかどうかをテストするには、次を実行します。

1. CA の **Manual User Signing & Encryption Certificates Enrollment** フォームを使用して、二重証明書に登録します。
2. 要求を送信します。エージェントサービスページにログインし、要求を承認します。
3. エンドエンティティーのページにログインし、証明書が発行されたかどうかを確認します。証明書のリストでは、連続するシリアル番号を持つ新しい証明書が 2 つあります。
4. Web ブラウザーに証明書をインポートします。
5. 鍵がアーカイブされたことを確認します。KRA のエージェントサービスページで、**Show completed requests** を選択します。キーが正常にアーカイブされた場合は、そのキーに関する情報が生成されます。キーが表示されない場合は、ログを確認して、問題を修正します。キーが正常にアーカイブされたら、ブラウザーウィンドウを閉じます。
6. 鍵を確認します。署名付きで暗号化された電子メールを送信します。メールを受信したら、メッセージを確認して開き、メッセージが署名されて暗号化されているかどうかを確認します。メッセージウィンドウの右上隅に、メッセージが署名されて暗号化されていることを示す

セキュリティーアイコンがあるはずです。

7. 証明書を削除します。暗号化された電子メールを再度確認します。メールクライアントはメッセージを復号できないはずです。

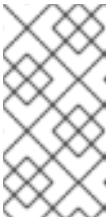
8. アーカイブされた鍵を正常に復元できるかどうかをテストします。

   a. KRA のエージェントサービスページを開き、**Recover Keys** リンクをクリックします。キーの所有者、シリアル番号、または公開鍵で鍵を検索します。キーが正常にアーカイブされた場合は、キー情報が表示されます。

   b. **Recover** をクリックします。

   c. 表示される形式で、復元する秘密鍵に対応する base-64 でエンコードされた証明書を入力します。この情報を取得するには CA を使用します。base-64 でエンコードされた証明書を指定してアーカイブされた鍵を検索する場合は、ここで証明書を指定する必要はありません。

   d. リカバリーの実行中にブラウザーセッションが閉じられるように **Async Recovery** チェックボックスが選択されていることを確認してください。



ヒント

非同期リカバリーはデフォルトであり、キーのリカバリーを実行するのに推奨される方法です。同期キーリカバリーを実行する場合、ブラウザーウィンドウはシャットダウンできず、リカバリープロセス中に KRA を停止できません。

   e. エージェントスキームに応じて、指定された数のエージェントがこの鍵のリカバリーを承認する必要があります。エージェントに、リカバリーキーを検索してもらい、開始された回復を承認してもらいます。

   f. すべてのエージェントがリカバリーを承認すると、次の画面は、証明書で PKCS #12 ファイルを暗号化するようにパスワードを要求します。

   g. 次の画面は、復元されたキーペアを含む PKCS #12 ブロブをダウンロードするリンクを返します。リンクに従い、blob をファイルに保存します。

重要

gcr-viewer ユーティリティーでブラウザーから直接 PKCS #12 ファイルを開くと、特定の状況で失敗する可能性があります。この問題を回避するには、**gcr-viewer** ファイルをダウンロードし、手動で開きます。

9. ブラウザーのデータベースに鍵を復元します。ブラウザーおよびメールクライアントに **.p12** ファイルをインポートします。

10. テストメールを開きます。メッセージは再度表示されます。

第5章 証明書の要求、登録、および管理

証明書は要求され、エンドユーザーに使用されます。証明書登録および更新は管理者に限定されていませんが、登録プロセスおよび更新プロセスを理解すると、「証明書プロファイルの設定」 で説明されているように、管理者が適切な証明書プロファイルを管理および作成でき、各証明書タイプに対して適切な認証方法 (10章 *証明書を登録するための認証* を参照) しやすくなります。

本章では、Certificate System 外で使用する証明書の要求、受信、および更新を説明します。Certificate System サブシステム証明書の要求および更新の詳細は、17章 *サブシステム証明書の管理* を参照してください。

5.1. 証明書の登録および更新について

登録 とは、証明書の要求および受信のプロセスです。登録プロセスの仕組みは、証明書の種類、キーペアの生成方法、および証明書自体の生成と承認の方法によってわずかに異なります。特定の方法が何であれ、高レベルでの証明書の登録には、同じ基本的な手順があります。

1. 証明書要求 (CSR) が生成されます。
2. 証明書要求が CA に送信されます。
3. 要求は、要求したエンティティーを認証し、それを提出するために使用された証明書プロファイルのルールを満たしていることを要求が確認することで検証されます。
4. リクエストが承認されている。
5. 要求側は新しい証明書を取得します。

証明書の有効期間が終了すると、証明書を更新できます。

5.2. 証明書署名リクエストの作成

従来は、証明書要求 (CSR) の生成には以下の方法が使用されます。

- コマンドラインユーティリティーを使用した CSR の生成
- 補助ブラウザー内での CSR の生成
- サーバーのインストーラーなど、アプリケーション内での CSR の生成

これらの方法の一部は CSR の直接送信をサポートしますが、含まれない場合があります。

RHCS 9.7 以降、サーバー側のキー生成がサポートされ、Firefox v69 以降や Chrome などの新しいバージョンのブラウザー内でのキー生成サポートの削除によってもたらされる不便を克服します。このため、本セクションでは、キー生成のブラウザーサポートについて説明しません。これらのブラウザーの古いバージョンが古い RHCS ドキュメントで指定されているように機能し続けるべきではないと信じる理由はありませんが。

通常、アプリケーションから生成された CSR は PKCS#10 の形式を取ります。それらが正しく生成されると、RHCS によりサポートされるはずです。

次のサブセクションでは、RHCS でサポートされている次の方法を説明します。

- コマンドラインユーティリティー
- サーバー側の鍵生成

5.2.1. コマンドラインユーティリティーを使用した CSR の生成

Red Hat Certificate System は、以下のユーティリティーを使用した CSR の作成をサポートします。

- **certutil**: PKCS #10 リクエストの作成に対応します。
- **PKCS10Client**: PKCS #10 リクエストの作成に対応します。
- **CRMFPopClient**: CRMF 要求の作成をサポートします。
- **pki client-cert-request**: PKCS#10 および CRMF リクエストの両方をサポートします。

次のセクションでは、これらのユーティリティーを機能豊富な登録プロファイルフレームワークで使用する方法の例をいくつか示します。

5.2.1.1. certutil を使用した CSR の作成

本セクションでは、**certutil** ユーティリティーを使用して CSR を作成する方法を説明します。

certutil の使用の詳細は、以下を参照してください。

- certutil(1) の man ページを参照してください。
- **certutil --help** コマンドの出力

5.2.1.1.1. certutil を使用した EC キーで CSR の作成

以下の手順は、**certutil** ユーティリティーを使用して Elliptic Curve(EC) キーペアと CSR を作成する方法を示しています。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. バイナリー CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ certutil -d . -R -k ec -q nistp256 -s "CN=subject_name" -o
/user_or_entity_database_directory/request-bin.csr
```

   プロンプトが表示されたら、必要な NSS データベースのパスワードを入力します。

   パラメーターの詳細は、certutil(1) の man ページを参照してください。

3. 作成したバイナリー形式の CSR を PEM 形式に変換します。

```
$ BtoA /user_or_entity_database_directory/request-bin.csr
/user_or_entity_database_directory/request.csr
```

4. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/request.csr
```

```
MIICbTCCAVUCAQAwKDEQMA4GA1UEChMHRXhhbXBsZTEUMBIGA1UEAxMLZXhhbXBs

...
```

これは、PKCS#10 PEM 証明書要求です。

5.2.1.1.2. certutil を使用したユーザー定義拡張による CSR の作成

以下の手順は、**certutil** ユーティリティーを使用してユーザー定義の拡張で CSR を作成する方法を示しています。

登録要求は、CA で定義された登録プロファイルにより制限されることに注意してください。???を参照してください。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. ユーザー定義の Extended Key Usage 拡張とユーザー定義の Key Usage 拡張で CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ certutil -d . -R -k rsa -g 1024 -s "CN=subject_name" --keyUsage
keyEncipherment,dataEncipherment,critical --extKeyUsage
timeStamp,msTrustListSign,critical -a -o /user_or_entity_database_directory/request.csr
```

プロンプトが表示されたら、必要な NSS データベースのパスワードを入力します。

パラメーターの詳細は、certutil(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/request.csr
  Certificate request generated by Netscape certutil
  Phone: (not specified)

  Common Name: user 4-2-1-2
  Email: (not specified)
  Organization: (not specified)
  State: (not specified)
  Country: (not specified)
```

これは、PKCS#10 PEM 証明書要求です。

5.2.1.2. PKCS10Client を使用した CSR の作成

本セクションでは、**PKCS10Client** ユーティリティーを使用して CSR を作成する方法を説明します。

PKCS10Client の使用に関する詳細は、以下を参照してください。

- PKCS10Client(1) の man ページを参照してください。
- **PKCS10Client --help** コマンドの出力

5.2.1.2.1. PKCS10Client を使用した CSR の作成

以下の手順では、**PKCS10Client** ユーティリティーを使用して Elliptic Curve (EC) キーペアと CSR を作成する方法を説明します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ PKCS10Client -d . -p NSS_password -a ec -c nistp256 -o
/user_or_entity_database_directory/example.csr -n "CN=subject_name"
```

パラメーターの詳細は、PKCS10Client(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.2.2. PKCS10Client を使用した SharedSecret ベースの CMC の CSR の作成

以下の手順では、**PKCS10Client** ユーティリティーを使用して、SharedSecret ベースの CMC 用の RSA 鍵ペアと CSR を作成する方法を説明します。これは、デフォルトでは **caFullCMCSharedTokenCert** プロファイルおよび **caECFullCMCSharedTokenCert** プロファイルによって処理される CMC 共有 Secret 認証方法にのみ使用します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ PKCS10Client -d . -p NSS_password -o /user_or_entity_database_directory/example.csr -y
true -n "CN=subject_name"
```

パラメーターの詳細は、PKCS10Client(1) の man ページを参照してください。

3. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.3. CRMFPopClient を使用した CSR の作成

Certificate Request Message Format (CRMF) は、CMC で受け入れられている CSR 形式であり、主要なアーカイブ情報を要求に安全に埋め込むことができます。

本セクションでは、**CRMFPopClient** ユーティリティーを使用して CSR を作成する方法を説明します。

CRMFPopClient の詳細な使用方法は、CRMFPopClient(1) の man ページを参照してください。

5.2.1.3.1. CRMFPopClient を使用したキー Archival を持つ CSR の作成

以下の手順では、**CRMFPopClient** ユーティリティーを使用して RSA キーペアと、鍵アーカイブオプションで CSR を作成する方法を説明します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. KRA トランスポート証明書を取得します。

```
$ pki ca-cert-find --name "DRM Transport Certificate"
  ---------------
  1 entries found
  ---------------
   Serial Number: 0x7
   Subject DN: CN=DRM Transport Certificate,O=EXAMPLE
   Status: VALID
   Type: X.509 version 3
   Key A    lgorithm: PKCS #1 RSA with 2048-bit key
   Not Valid Before: Thu Oct 22 18:26:11 CEST 2015
   Not Valid After: Wed Oct 11 18:26:11 CEST 2017
   Issued On: Thu Oct 22 18:26:11 CEST 2015
   Issued By: caadmin
  ----------------------------
  Number of entries returned 1
```

3. KRA トランスポート証明書をエクスポートします。

```
$ pki ca-cert-show 0x7 --output kra.transport
```

4. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b
kra.transport -w "AES/CBC/PKCS5Padding" -v -o
/user_or_entity_database_directory/example.csr
```

Elliptic Curve (EC) キーペアと CSR を作成するには、**-a ec -t false** オプションをコマンドに渡します。

パラメーターの詳細は、CRMFPopClient(1) の man ページを参照してください。

5. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.3.2. CRMFPopClient を使用した SharedSecret ベースの CMC の CSR の作成

以下の手順では、**CRMFPopClient** ユーティリティーを使用して、SharedSecret ベースの CMC 用の RSA 鍵ペアと CSR を作成する方法を説明します。これは、デフォルトでは **caFullCMCSharedTokenCert** プロファイルおよび **caECFullCMCSharedTokenCert** プロファイルによって処理される CMC 共有 Secret 認証方法にのみ使用します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /user_or_entity_database_directory/
```

2. KRA トランスポート証明書を取得します。

```
$ pki ca-cert-find --name "DRM Transport Certificate"
 ---------------
 1 entries found
 ---------------
  Serial Number: 0x7
  Subject DN: CN=DRM Transport Certificate,O=EXAMPLE
  Status: VALID
  Type: X.509 version 3
  Key A    lgorithm: PKCS #1 RSA with 2048-bit key
  Not Valid Before: Thu Oct 22 18:26:11 CEST 2015
  Not Valid After: Wed Oct 11 18:26:11 CEST 2017
  Issued On: Thu Oct 22 18:26:11 CEST 2015
  Issued By: caadmin
 ----------------------------
 Number of entries returned 1
```

3. KRA トランスポート証明書をエクスポートします。

```
$ pki ca-cert-show 0x7 --output kra.transport
```

4. CSR を作成し、これを ***/user_or_entity_database_directory/request.csr*** ファイルに保存します。

```
$ CRMFPopClient -d . -p password -n "cn=subject_name" -q POP_SUCCESS -b
kra.transport -w "AES/CBC/PKCS5Padding" -y -v -o
/user_or_entity_database_directory/example.csr
```

EC キーペアと CSR を作成するには、コマンドに **-a ec -t false** オプションを渡します。

パラメーターの詳細は、**CRMFPopClient --help** コマンドの出力を参照してください。

5. 必要に応じて、CSR ファイルが正しいことを確認します。

```
$ cat /user_or_entity_database_directory/example.csr
 -----BEGIN CERTIFICATE REQUEST-----
 MIICzzCCAbcCAQAwgYkx
 ...
 -----END CERTIFICATE REQUEST-----
```

5.2.1.4. PKI CLI での client-cert-request を使用した CSR の作成

pkiコマンドラインツールは、**client-cert-request** コマンドで使用して CSR を生成することもできます。ただし、前述のツールとは異なり、pki で 生成された CSR は CA に直接送信されます。PKCS#10 または CRMF 要求の両方を生成できます。

PKCS#10 要求の生成例:

```
pki -d user token db directory -P https -p 8443 -h host.test.com -c user token db passwd client-cert-request "uid=test2" --length 4096 --type pkcs10
```

CRMF リクエストの生成例:

```
pki -d user token db directory -P https -p 8443 -h host.test.com -c user token db passwd client-cert-request "uid=test2" --length 4096 --type crmf
```

成功するとリクエスト ID が返されます。

要求が送信されると、エージェントは pki **ca-cert-request-approve** コマンドを使用して承認できます。

以下に例を示します。

```
pki -d agent token db directory -P https -p 8443 -h host.test.com -c agent token db passwd -n <CA agent cert nickname> ca-cert-request-approve request id
```

詳細は、**pki client-cert-request --help** コマンドを実行して man ページを参照してください。

5.2.2. サーバー側の鍵生成を使用した CSR の生成

Firefox v69 以降や Chrome など、多くの新しいバージョンのブラウザーでは、PKI キーを生成する機能と、キーアーカイブ用の CRMF のサポートが削除されています。RHEL では、**CRMFPopClient** (**CRMFPopClient--help** を参照) または **pki** (**pki client-cert-request --help** を参照) などの CLI を回避策として使用することができます。

サーバー側の Keygen の登録は、トークンキー管理システム (TMS) の導入以来、長い間行われてきました。このシステムでは、キーをスマートカードでローカルに生成するのではなく、KRA で生成できます。Red Hat Certificate System では、ブラウザーのキー生成の問題を解決するための同様のメカニズムが導入されました。鍵はサーバーで生成され (特に KRA で)、PKCS#12 のクライアントに安全に転送されます。

注記

暗号化証明書にのみサーバー側 Keygen メカニズムを使用することが強く推奨されます。

5.2.2.1. 主な機能

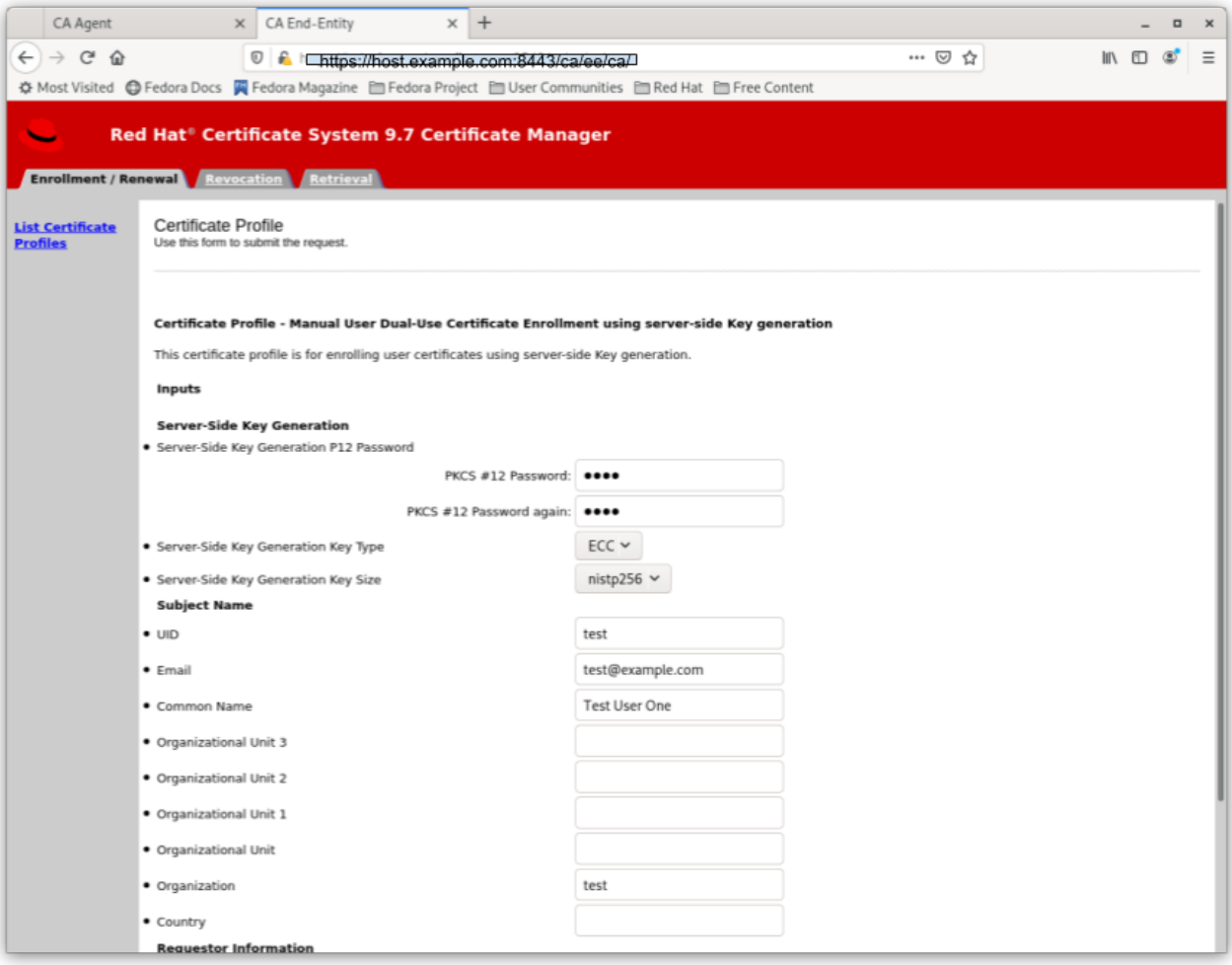
- 証明書要求キーは KRA で生成されます (注: CA と連携するには、KRA をインストールする必要があります)。
- プロファイルのデフォルトプラグイン **serverKeygenUserKeyDefaultImpl** は、キーアーカイブ (つまり **enableArchival** パラメーター) を有効または無効にするための選択を提供します。
- RSA 鍵と EC 鍵の両方のサポート
- 手動 (エージェント) 承認と自動承認 (ディレクトリーパスワードベースなど) の両方のサポート

5.2.2.2. Server-Side Keygen を使用した証明書の登録

デフォルトの Sever-Side Keygen 登録プロファイルは、EE ページの **List Certificate Profiles** タブにあります。

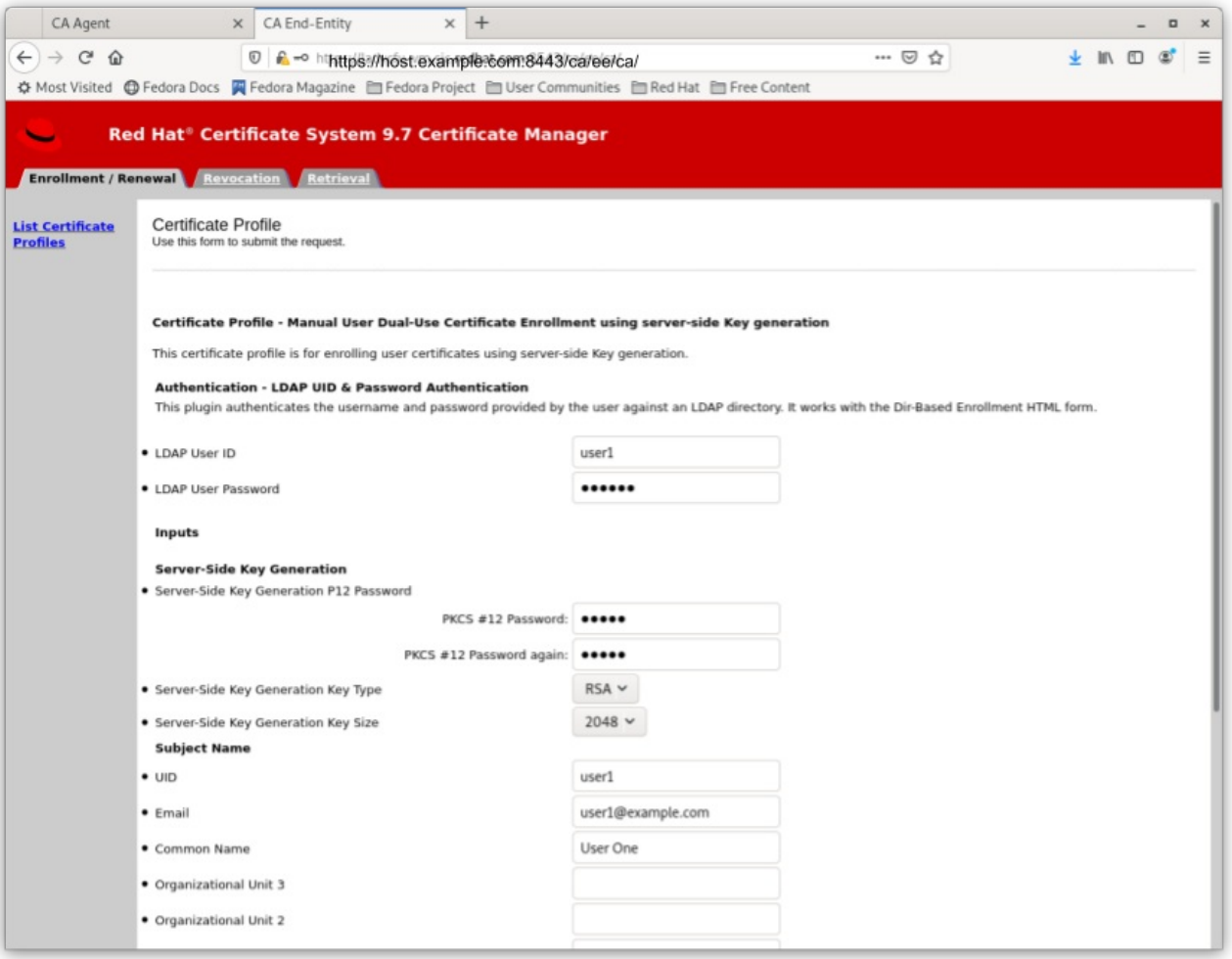
サーバー側の鍵の生成を使用した手動ユーザーのデュアル使用証明書登録

図5.1 エージェントの手動による承認を必要とするサーバー側のキータイプの登録



サーバー側の鍵生成を使用したディレクトリー認証ユーザーのデュアル使用証明書の登録

図5.2 LDAP の uid/pwd 認証が正常に実行されると自動的に承認される サーバー側のキータイプの登録



リクエストの承認方法に関係なく、Server-Side Keygen Enrollment メカニズムでは、エンドエンティティーユーザーが PKCS#12 パッケージのパスワードを入力する必要があります。このパスワードには、発行された証明書と、発行後にサーバーによって生成された暗号化された秘密鍵が含まれます。

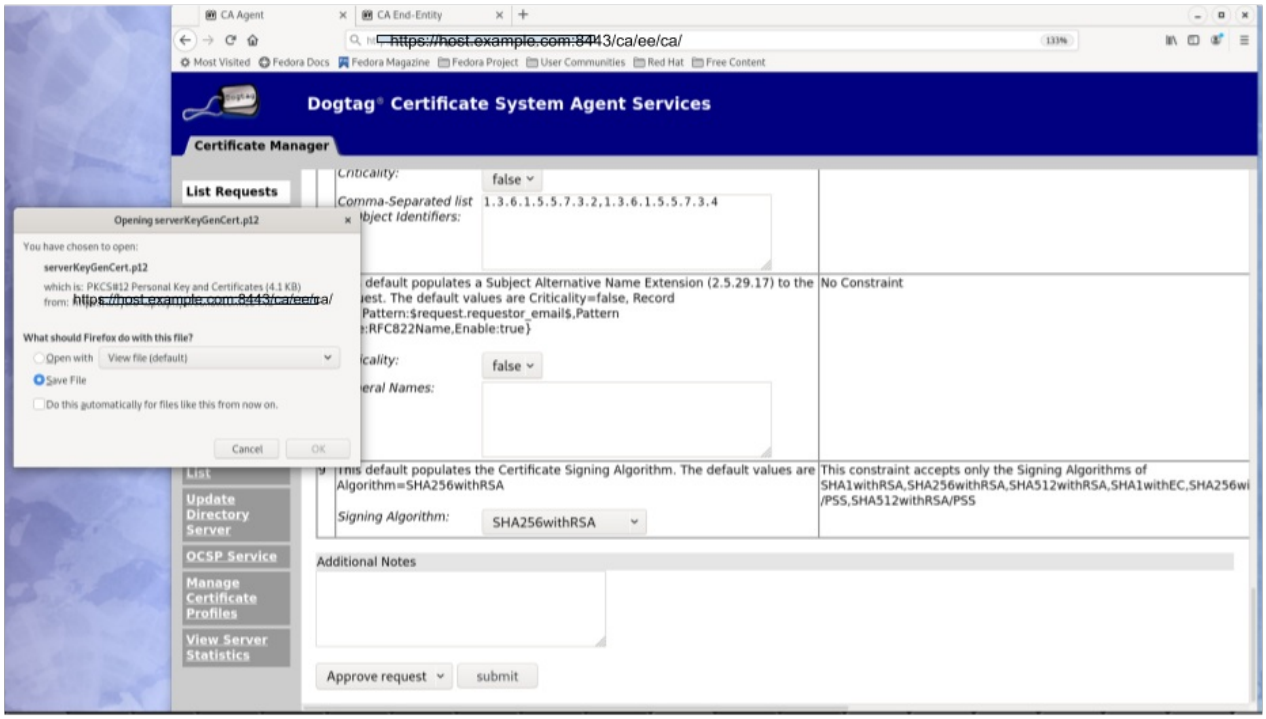
重要

パスワードは共有しないでください。CA や KRA のエージェントでさえありません。

登録要求が承認されると、PKCS#12 パッケージが生成され、以下が行われます。

- 手動承認の場合、PKCS#12 ファイルは要求を承認する CA エージェントに返されます。その後、エージェントは PKCS#12 ファイルをユーザーに転送することが期待されます。
- 自動承認の場合、PKCS#12 ファイルはリクエストを送信したユーザーに返されます。

図5.3 エージェントによる手動による登録



PKCS#12 ファイルを受け取ったら、アプリケーションごとにこのファイルをユーザーの内部証明書/キーデータベースに **pkcs12util** インポートするなどの CLI を使用できます。たとえば、ユーザーの Firefox nss データベースです。

5.2.2.3. キーリカバリー

証明書登録プロファイルで **enableArchival** パラメーターが **true** に設定されている場合、Server-Side Keygen 登録時に秘密鍵がアーカイブされます。その後、アーカイブされた秘密鍵は、認定 KRA エージェントにより復元できます。

5.2.2.4. 追加情報

5.2.2.4.1. KRA 要求レコード

注記

このメカニズムの性質上、プロファイルで **enableArchival** パラメーターが **true** に設定されている場合、Server-Side keygen 要求ごとに 2 つの KRA 要求レコードがあります。

- 要求タイプ **asymkeyGenRequest** の 1 つ

  この要求タイプは、KRA エージェントページの **List Requests** を使用してフィルターすることはできません。**Show All Requests** を選択して、一覧を表示できます。

- 要求タイプの **リカバリー** の場合

5.2.2.4.2. 監査レコード

有効にした場合には、以下の監査レコードを確認することができます。

CA

- SERVER_SIDE_KEYGEN_ENROLL_KEYGEN_REQUEST
- SERVER_SIDE_KEYGEN_ENROLL_KEY_RETRIEVAL_REQUEST

KRA

- SERVER_SIDE_KEYGEN_ENROLL_KEYGEN_REQUEST_PROCESSED
- SERVER_SIDE_KEYGEN_ENROLL_KEY_RETRIEVAL_REQUEST_PROCESSED (まだ実装されていません)

5.3. 証明書の要求および受信

「証明書の登録および更新について」 で説明されているように、CSR が生成されたら、発行用に CA に送信する必要があります。「証明書署名リクエストの作成」 で説明されている方法のいくつかが、CSR を CA に直接送信しますが、CSR を別のステップで送信する必要がある場合もあります。これは、ユーザーが実行するか、エージェントが事前に署名することができます。

本セクションでは、RHCS CA でサポートされる別の送信手順を説明します。

- 「End-Entities ページでの証明書の要求および受信」
- 「CMC を使用した証明書要求の送信」

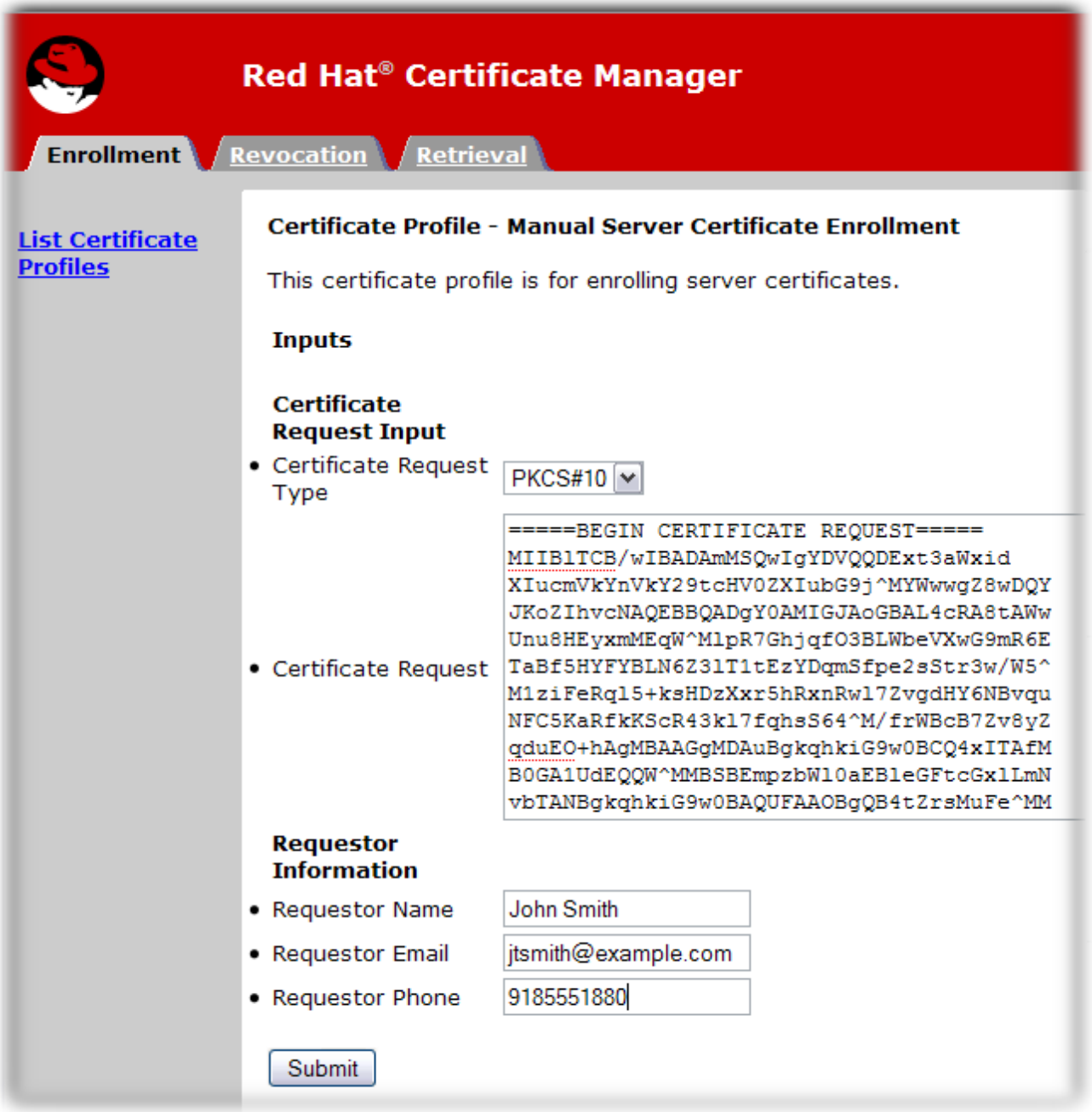
5.3.1. End-Entities ページでの証明書の要求および受信

CA エンドエンティティーポータル (つまり、https://*host.domain*:*port*#/ca/ee/ca)) で、エンドエンティティーは、**Enrollment**/**Renewal** タブの該当する各登録プロファイルに表示される HTML 登録フォームを使用して、証明書要求を送信できます (CSR。CSR の生成方法は 「証明書署名リクエストの作成」 を参照)。

このセクションでは、マーカー行 -----BEGIN NEW CERTIFICATE REQUEST----- および -----END NEW CERTIFICATE REQUEST----- を含む Base64 エンコード形式の CSR があることを前提としています。

デフォルトの登録プロファイルの多くは、Base64 でエンコードされた CSR に貼り付けることができる **証明書要求** テキストボックスと、**証明書要求タイプ** の選択ドロップダウンリストを提供します。

証明書の登録フォームで、必要な情報を入力します。



標準の要件は以下のとおりです。

- **証明書要求のタイプ**。これは PKCS#10 または CRMF です。サブシステム管理コンソールを介して作成された証明書要求は PKCS#10 です。**certutil** ツールを通じて作成されたものやその他のユーティリティーは通常 PKCS #10 です。

- **証明書要求**。**-----BEGIN NEW CERTIFICATE REQUEST-----** および **-----END NEW CERTIFICATE REQUEST-----** マーカー行を含む base-64 でエンコードされた BLOB を貼り付けます。

- **Requester Name**。これは、証明書を要求するユーザーの共通名です。

- **Requester Email**。これは、要求側のメールアドレスです。エージェントまたは CA システムは、このアドレスを使用して、証明書を発行する際に要求側に接続します。たとえば、**jdoe@someCompany.com** です。

- **Requester Phone**。これは、要求側の連絡先番号です。

送信されたリクエストは、エージェントの承認のためにキューに置かれます。エージェントは、証明書要求を処理し、承認する必要があります。

注記

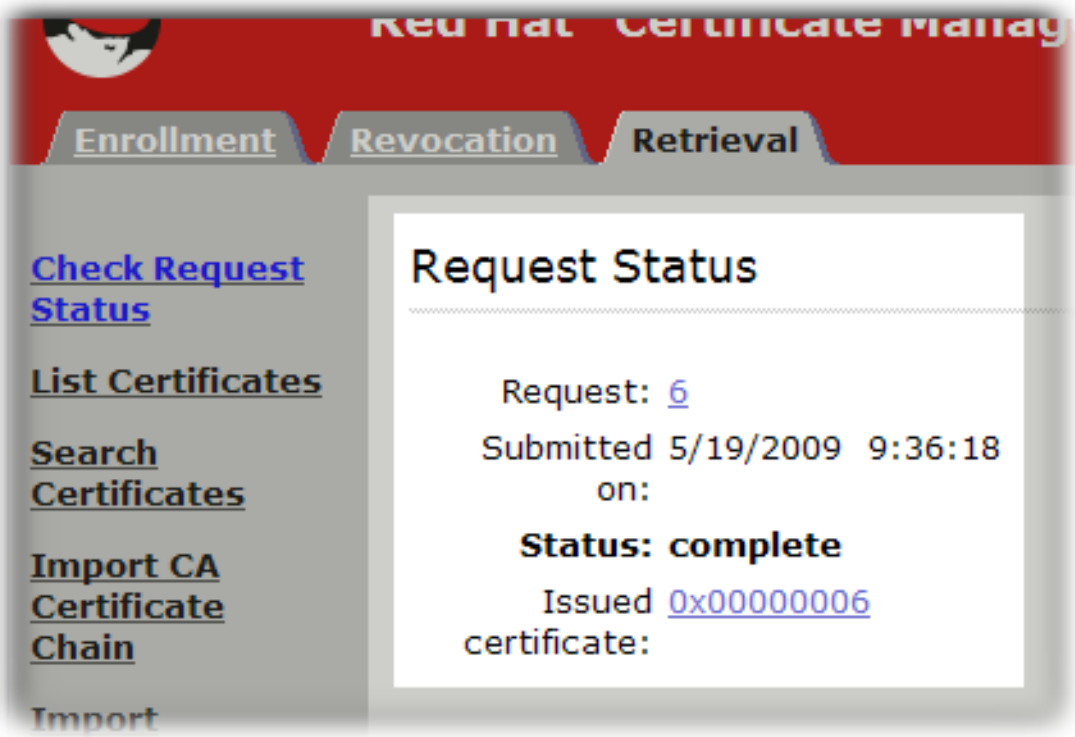
登録プロファイルによっては、Red Hat Certificate System が提供する LDAP uid/pwd 認証メソッドを使用することで、自動承認を行うことがあります。これらのプロファイルによる登録では、次のセクションに手動でエージェントの承認は必要ありません。サポートされる承認方法は、*10章 証明書を登録するための認証* を参照してください。

手動承認の場合は、証明書が承認および生成されると、証明書を取得できます。

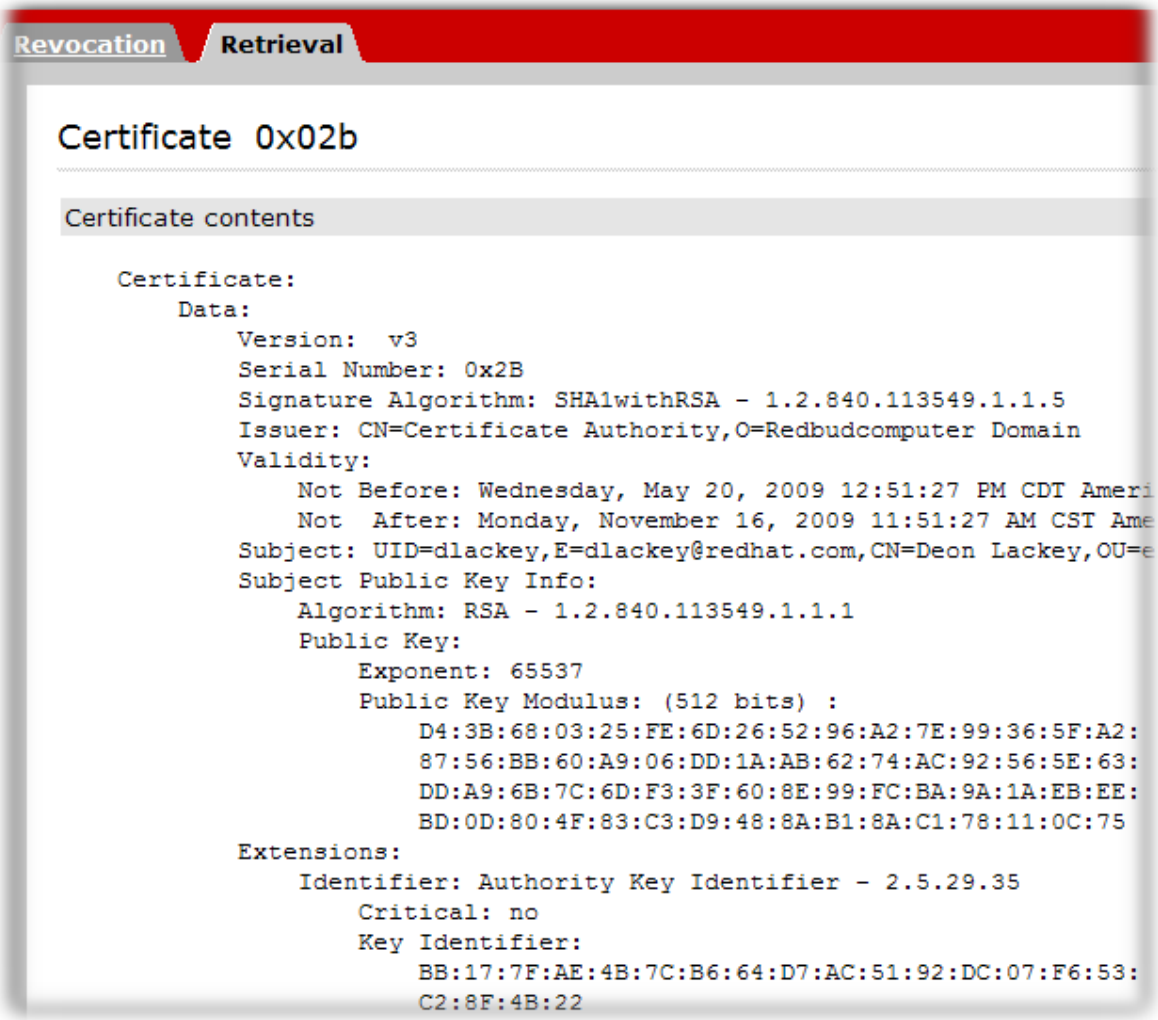
1. Certificate Manager の end-entities ページを開きます。以下に例を示します。

   ```
   https://server.example.com:8443/ca/ee/ca
   ```

2. **Retrieval** タブをクリックします。
3. 証明書要求の送信時に作成された要求 ID 番号を入力し、**Submit** をクリックします。
4. 次のページには、証明書要求のステータスが表示されます。ステータスが **complete** すると、証明書へのリンクがあります。**Issued certificate** のリンクをクリックします。



5. 新しい証明書情報は、pretty-print 形式、base-64 エンコード形式、および PKCS #7 形式で表示されます。



このページでは、以下のアクションを実行できます。

- この証明書をサーバーまたは他のアプリケーションにインストールするには、base-64 でエンコードされた証明書が含まれている **Installing This Certificate in a Server** セクションまで下にスクロールします。

6. base-64 でエンコードされた証明書 (マーカー行 **-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** を含む) をテキストファイルにコピーします。テキストファイルを保存し、これを使用して秘密鍵があるエンティティーのセキュリティーモジュールに証明書のコピーを保存します。「ユーザーの作成」を参照してください。

5.4. 証明書の更新

本セクションでは、証明書の更新方法を説明します。証明書の更新の設定方法は 「更新を有効にするためのプロファイルの設定」 を参照してください。

証明書の更新は、元の証明書と同じ目的で使用されるプロパティーを使用して、証明書を再生成します。一般的には、更新には 2 つのタイプがあります。

- **同じキーの更新** は、証明書の元のキー、プロファイル、および要求を受け取り、同じキーを使用して、新しい有効期間と有効期限で新しい証明書を再作成します。これは、以下のいずれかの方法で実行できます。
  - 元のプロファイルから元の証明書要求 (CSR) の再送信、または
  - certutil などのサポートツールを使用した元のキーで CSR の再生成

- 証明書の **キーを再生成** するには、同じ情報を使用して証明書要求を再生成する必要があるため、新しいキーペアが生成されます。その後、CSR は元のプロファイルを介して送信されます。

5.4.1. 同じキーの更新

5.4.1.1. CSR の再利用

エンドエンティティーポータルには、同じキー更新に対する承認メソッドが 3 つあります。

- エージェントが承認した方法では、更新する証明書のシリアル番号を送信する必要があります。この方法では、CA エージェントの承認が必要になります。
- ディレクトリーベースの更新では、更新する証明書のシリアル番号を送信する必要があり、CA は現在の証明書ディレクトリーエントリーから情報を取得します。ldap uid/pwd が正常に認証されると、証明書は自動的に承認されます。
- 証明書ベースの更新は、ブラウザーデータベースの証明書を使用して認証し、同じ証明書を再発行します。

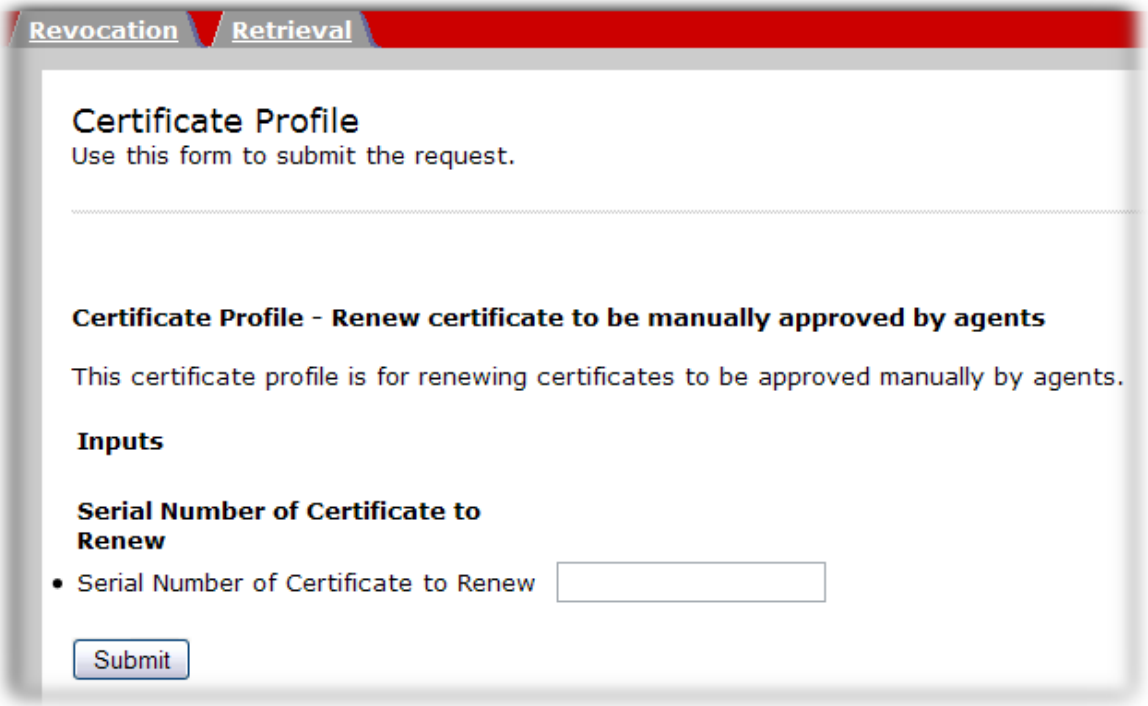
5.4.1.1.1. エージェントによる承認またはディレクトリーベースの更新

場合によっては、CA エージェントによって、またはユーザーディレクトリーのログイン情報を提供することによって、証明書の更新要求を手動で承認する必要があります。

1. 証明書 (またはそのクローン) の CA のエンドエンティティーサービスページを開きます。

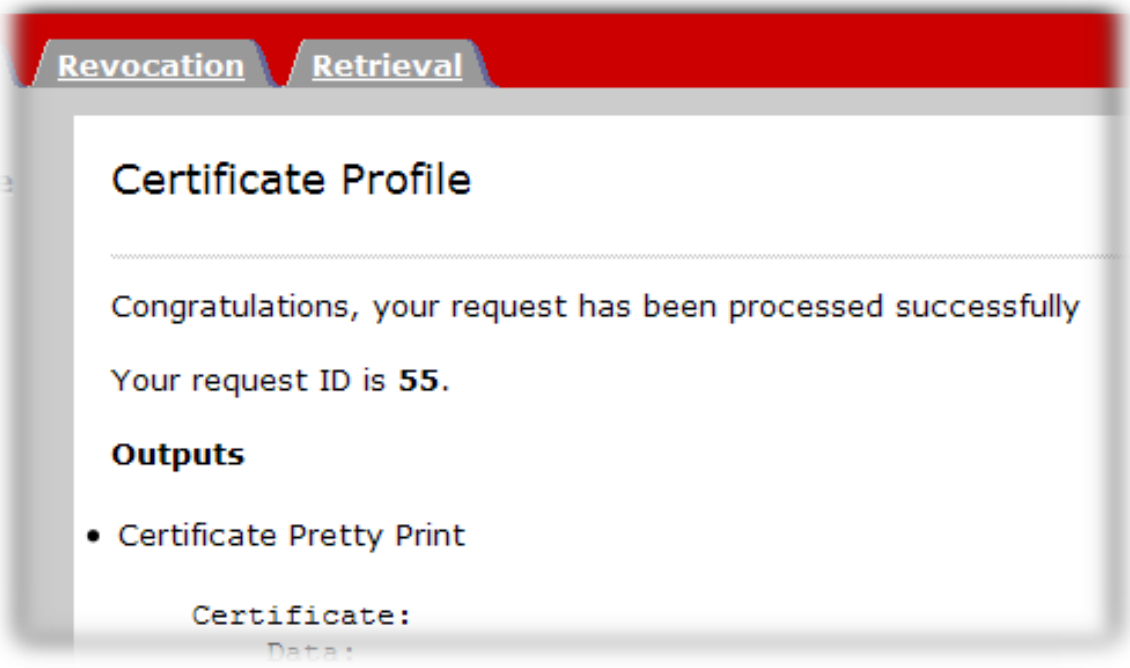
   ```
   https://server.example.com:8443/ca/ee/ca
   ```

2. 使用する更新フォームの名前をクリックします。
3. 更新する証明書のシリアル番号を入力します。これは、10 進数または 16 進数の形式にすることができます。



4. 更新ボタンをクリックします。

5. リクエストが送信されます。ディレクトリーベースの更新では、更新された証明書が自動的に返されます。それ以外の場合、更新リクエストはエージェントにより承認されます。



5.4.1.1.2. 証明書ベースの更新

ユーザー証明書によってはブラウザーに直接保存されるため、更新フォームによっては、更新する証明書について、ブラウザーの証明書データベースを確認するだけです。証明書を更新できる場合は、CA が自動的に承認され、再発行されます。

重要

更新中の証明書の有効期限が **すでに** 切れている場合は、証明書ベースの更新には使用できない可能性があります。ブラウザークライアントは、期限切れの証明書での SSL クライアント認証を許可しない可能性があります。

この場合には、他の更新方法のいずれかを使用して証明書を更新する必要があります。

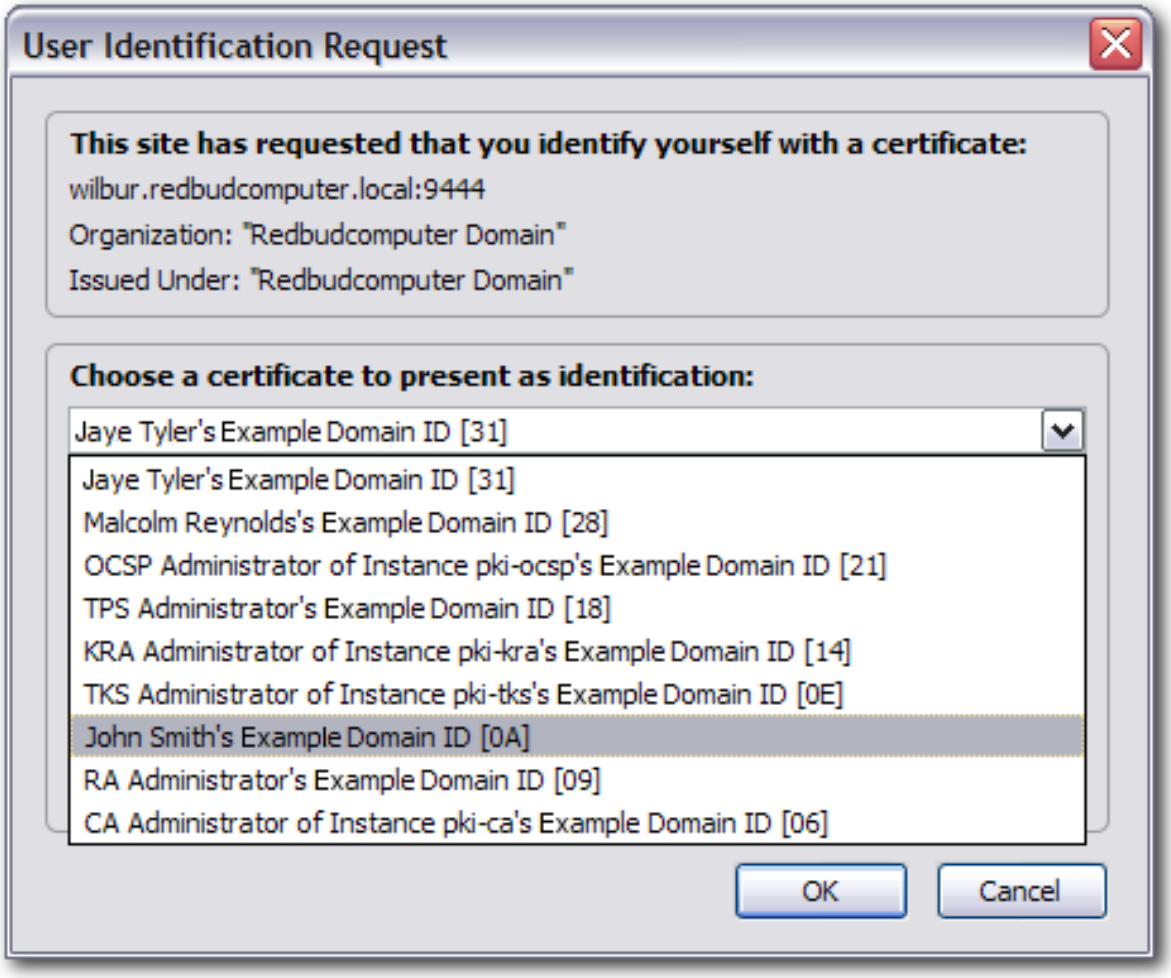
1. 証明書 (またはそのクローン) の CA のエンドエンティティーサービスページを開きます。

   ```
   https://server.example.com:8443/ca/ee/ca
   ```

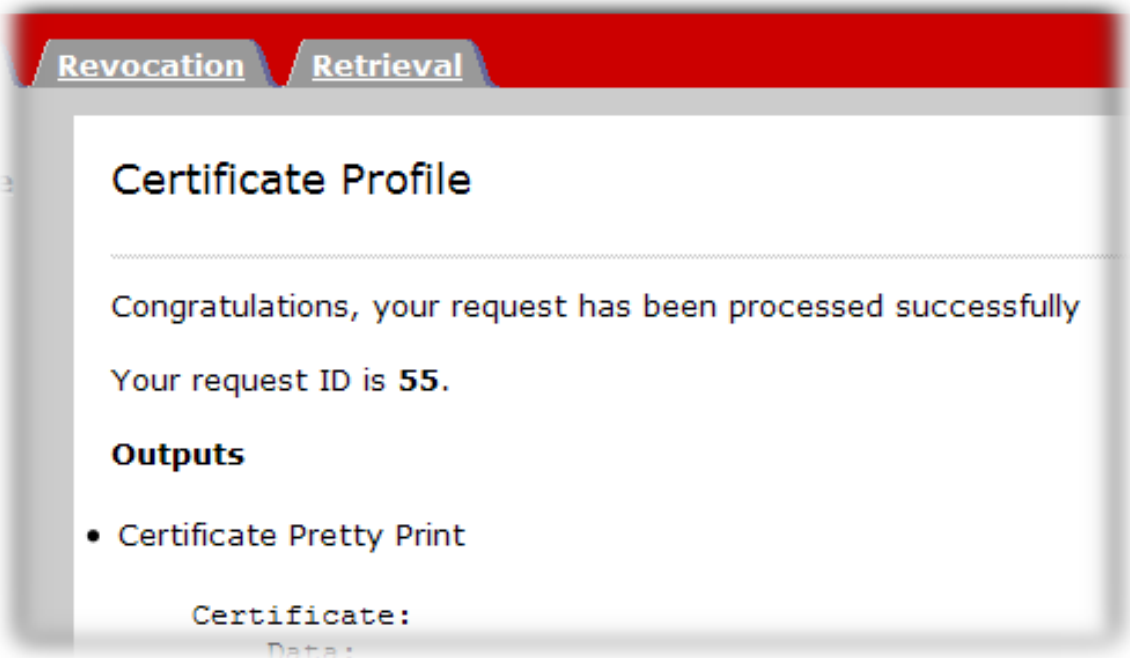
2. 使用する更新フォームの名前をクリックします。

3. 入力フィールドがないため、**Renew** ボタンをクリックします。

4. プロンプトが表示されたら、更新する証明書を選択します。



5. 要求が送信され、更新された証明書が自動的に返されます。



5.4.1.2. 同じキーを持つ CSR を生成して更新

元の CSR が利用できない場合があります。この **certutil** ツールを使用して、キーペアが NSS データベースにある場合に、同じキーで CSR を再生成できます。これは、次の手順で実行できます。

1. NSS データベースで、対応するキー ID を検索します。

```
Certutil -d <nssdb dir> -K
```

2. 特定のキーを使用して CSR を生成します。

```
Certutil -d <nssdb dir> -R -k <key id> -s <subject DN> -o <CSR output file>
```

または、**keyid** の代わりに、キーが NSS データベースの証明書に関連付けられている場合は、**ニックネーム** を使用できます。

- 既存のニックネームを使用して CSR を生成します。

```
Certutil -d <nssdb dir> -R -k <nickname> -s <subject DN> -o <CSR output file>
```

5.4.2. 証明書のキー変更による更新

キーの再生成 による更新は、基本的に古い証明書と同じ情報で新しい CSR を生成するため、「証明書署名リクエストの作成」 で説明されている方法のいずれかに従ってください。古い証明書と同じ情報を入力します。

5.5. CMC を使用した証明書要求の送信

このセクションでは、CMS (Certificate Management over CMS) を使用して証明書を登録する手順を説明します。

CMC を使用して証明書を登録する設定とワークフローの一般的な情報は、以下を参照してください。

- 『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『CMC の設定』 セクションを参照してください。
- 『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『CMC を使用した登録』 セクション
- CMCRequest(1) の man ページを参照してください。
- CMCResponse(1) の man ページを参照してください。

CMC の登録は、さまざまなシナリオの要件を満たすためにさまざまな方法で可能です。「CMC 登録プロセス」 は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『CMC を使用した登録』 セクションを補足します。さらに、「実用的な CMC 登録シナリオ」 セクションを使用すると、管理者はどのメカニズムをどのシナリオで使用するかを決定できます。

5.5.1. CMC 登録の使用

CMC 登録により、登録クライアントは認証に CMCAuth プラグインを使用できます。これにより、証明書要求はエージェント証明書で事前署名されます。Certificate Manager は、エージェント証明書で署名した有効な要求を受け取れると、証明書を自動的に発行します。

注記

CMC 登録はデフォルトで有効になっています。設定が変更されていない限り、CMC 登録認証プラグインまたはプロファイルを有効にする必要はありません。

CMCAuth 認証プラグインは、クライアントに CMC 失効も提供します。CMC の失効により、クライアントはエージェント証明書によって署名された証明書要求を取得し、そのような要求を Certificate Manager に送信できます。Certificate Manager は、エージェント証明書で署名した有効な要求を受け取ると、証明書を自動的に取り消します。**CMCRevoke** コマンドラインツールを使用して、CMC 失効を作成できます。**CMCRevoke** の詳細は、「CMC 失効の実行」 を参照してください。

CMC リクエストは、ブラウザーのエンドエンティティーフォームから、または **HttpClient** などのツールを使用して送信して、適切なプロファイルにリクエストを投稿できます。この **CMCRequest** ツールは、署名済み証明書要求を生成し、**HttpClient** ツールまたはブラウザーのエンドエンティティーフォームを使用して、証明書を自動的かつ即座に登録および受信します。

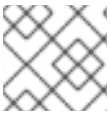
CMCRequest ツールには簡単なコマンド構文があり、**.cfg** 入力ファイルに指定されるすべての設定が設定されます。

```
CMCRequest /path/to/file.cfg
```

以下の構文で、**CMCEnroll** ツールを使用して 1 回の登録を作成することもできます。

```
CMCEnroll -d /agent's/certificate/directory -h password -n cert_nickname -r certrequest.file -p
certDB_passwd [-c "comment"]
```

これらのツールの詳細は、**CMCEnroll(1)** の man ページで説明されています。



注記

引用符で囲まれたスペースを含む値を囲みます。

5.5.1.1. CMCEnroll のテスト

1. **certutil** ツールを使用して証明書要求を作成します。
2. PKCS #10 ASCII 出力をテキストファイルにコピーします。
3. CMCEnroll ユーティリティーを実行します。

   たとえば、入力ファイルが **request34.txt** を呼び出すと、エージェント証明書はブラウザーデータベースに保存され、エージェント証明書の証明書の一般名は **CertificateManagerAgentsCert** で、および証明書データベースのパスワードは **secret** で、コマンドは次のとおりです。

   ```
   CMCEnroll -d ~jsmith/.mozilla/firefox/1234.jsmith -n "CertificateManagerAgentsCert" -r
   /export/requests/request34.txt -p secret
   ```

   このコマンドの出力は、ファイル名に加えられた **.out** で同じファイル名のファイルに保存されます。
4. エンドエンティティーを通じて署名済み証明書を提出します。

   a. エンドエンティティーを開きます。

   ```
   https://server.example.com:8443/ca/ee/ca
   ```

   b. 証明書プロファイルのリストから CMC 登録フォームを選択します。

c. 出力ファイルの内容をこの形式の **Certificate Request** テキスト領域に貼り付けます。

d. 貼り付けられたコンテンツから **-----BEGIN NEW CERTIFICATE REQUEST-----** および **----END NEW CERTIFICATE REQUEST-----** を削除します。

e. 連絡先情報を入力して、フォームに入力します。

5. 証明書は即座に処理され、返されます。

6. エージェントページを使用して、新しい証明書を検索します。

5.5.2. CMC 登録プロセス

CMC を使用して証明書を要求および発行するには、次の一般的な手順を使用します。

1. Certificate Signing Request (CSR) を、以下のいずれかの形式で作成します。

   - PKCS #10 形式
   - Certificate Request Message Format (CRMF) 形式

   これらの形式で CSR を作成する方法は、「証明書署名リクエストの作成」 を参照してください。

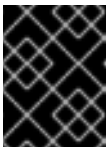
2. 管理証明書をクライアントの NSS データベースにインポートします。以下に例を示します。

   - 以下のコマンドを実行して、**.p12** ファイルから管理クライアント証明書を抽出します。

     ```
     $ openssl pkcs12 -in /root/.dogtag/instance/ca_admin_cert.p12 -clcerts -nodes -nokeys -out /root/.dogtag/instance/ca_admin_cert.crt
     ```

   - 『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『証明書/キー暗号化トークンの管理』セクションに従って、管理クライアント証明書の検証およびインポートを行います。

     ```
     $ PKICertImport -d . -n "CA Admin - Client Certificate" -t ",," -a -i /root/.dogtag/instance/ca_admin_cert.crt -u C
     ```

     重要

     CA 管理クライアント証明書をインポートする前に、中間証明書とルート CA 証明書がインポートされていることを確認します。

   - 証明書に関連付けられた秘密鍵をインポートします。

     ```
     $ pki -c password pkcs12-import --pkcs12-file /root/.dogtag/instance/ca_admin_cert.p12 --pkcs12-password-file /root/.dogtag/instance/ca/pkcs12_password.conf
     ```

3. 以下の内容で、**/home/user_name/cmc-request.cfg** などの CMC 要求用の設定ファイルを作成します。

   ```
   # NSS database directory where CA agent certificate is stored
   dbdir=/home/user_name/.dogtag/nssdb/
   ```

```
# NSS database password
password=password

# Token name (default is internal)
tokenname=internal

# Nickname for signing certificate
nickname=subsystem_admin

# Request format: pkcs10 or crmf
format=pkcs10

# Total number of PKCS10/CRMF requests
numRequests=1

# Path to the PKCS10/CRMF request
# The content must be in Base-64 encoded format.
# Multiple files are supported. They must be separated by space.
input=/home/user_name/file.csr

# Path for the CMC request
output=/home/user_name/cmc-request.bin
```

詳細は、CMCRequest(1) の man ページを参照してください。

4. CMC 要求を作成します。

```
$ CMCRequest /home/user_name/cmc-request.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

5. **/home/user_name/cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
# PKI server host name
host=server.example.com

# PKI server port number
port=8443

# Use secure connection
secure=true

# Use client authentication
clientmode=true

# NSS database directory where the CA agent certificate is stored.
dbdir=/home/user_name/.dogtag/nssdb/

# NSS database password
password=password

# Token name (default: internal)
```

```
tokenname=internal

# Nickname of signing certificate
nickname=subsystem_admin

# Path for the CMC request
input=/home/user_name/cmc-request.bin

# Path for the CMC response
output=/home/user_name/cmc-response.bin
```

重要

nickname パラメーターで指定された証明書のニックネームは、CMC 要求で以前使用された内容と一致させる必要があります。

6. 要求する証明書のタイプに応じて、前の手順で作成した設定ファイルに次のパラメーターを追加します。

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=profile_name
```

CA 署名証明書の場合の例を以下に示します。

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCcaCert
```

重要

エージェントが次のステップで CMC 要求を送信する場合は、このパラメーターで指定したプロファイルは **CMCAuth** 認証プラグインを使用する必要があります。ユーザーが作成した登録では、プロファイルは **CMCUserSignedAuth** プラグインを使用する必要があります。詳細は、「CMC 認証プラグイン」 を参照してください。

7. CMC 要求を CA に送信します。

```
$ HttpClient /home/user_name/cmc-submit.cfg
```

8. CMC の応答を PKCS #7 証明書チェーンに変換するには、**CMCResponse** ユーティリティーの ***-i*** パラメーターに CMC レスポンスファイルを渡します。以下に例を示します。

```
$ CMCResponse -i /home/user_name/cmc-response.bin -o /home/user_name/cert_chain.crt
```

5.5.3. 実用的な CMC 登録シナリオ

本セクションでは、CA 管理者がどの状況でどの CMC メソッドを使用するかを決定できるようにするための、頻繁な実際の使用シナリオとそのワークフローを説明します。

CMC を使用して証明書を登録する一般的なプロセスは、「CMC 登録プロセス」 を参照してください。

5.5.3.1. システム証明書およびサーバー証明書の取得

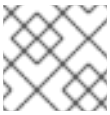
LDAP や Web サーバーなどのサービスで TLS サーバー証明書が必要な場合、このサーバーの管理者はサービスのドキュメントに基づいて CSR を作成し、承認のために CA のエージェントに送信します。このプロセスには、「CMC 登録プロセス」 で説明されている手順を使用します。また、以下の要件を考慮してください。

登録プロファイル

エージェントは、「CMC 認証プラグイン」 にリストされている既存の CMC プロファイルのいずれかを使用する必要があります。または、**CMCAuth** 認証メカニズムを使用するカスタムプロファイルを作成します。

CMC 署名証明書

システム証明書の場合、CA エージェントは CMC 要求を生成して署名する必要があります。そのためには、**CMCRequest** 設定ファイルの ***nickname*** パラメーターを CA エージェントのニックネームに設定します。



注記

CA エージェントは、独自の秘密鍵にアクセスできるようにする必要があります。

HttpClient TLS Client Nickname

HttpClient の設定ファイル内で TLS クライアント認証に関するユーティリティーの設定ファイルに対して、**CMCRequest** ユーティリティー設定ファイルへのサインインに同じ証明書を使用します。

HttpClient *servlet* パラメーター

HttpClient ユーティリティーに渡される設定ファイルの ***servlet*** では、要求を処理する CMC サーブレットおよび登録プロファイルが参照されます。

要求する証明書のタイプに応じて、直前の手順で作成した設定ファイルに以下のエントリーのいずれかを追加します。

- CA 署名証明書の場合:

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCcaCert
```

- KRA トランスポート証明書の場合:

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCkraTransportCert
```

- OCSP 署名証明書の場合:

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCocspCert
```

- 監査署名証明書の場合:

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCauditSigningCert
```

- サブシステム証明書の場合:
  - RSA 証明書の場合:

```
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCsubsystemCert
```

-

- ECC 証明書の場合:

  servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCECCsubsystemCert

- TLS サーバー証明書の場合:
  - RSA 証明書の場合:

    servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCserverCert

  - ECC 証明書の場合:

    servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCECCserverCert

- 管理証明書の場合:

  servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caFullCMCUserCert

詳細は以下を参照してください。

- エージェントが CSR を事前署名する場合、エージェントは識別のために CSR を調べるため、Proof of Identification が確立されたと見なされます。追加の CMC 固有の識別証明は必要ありません。
- PKCS #10 ファイルはすでに Proof of Possession (POP) 情報を提供し、追加の Proof of Possession (POP) は必要ありません。
- エージェントの事前承認済みリクエストでは、識別はエージェントによってチェックされるため、**PopLinkWittnessV2** 機能を無効にする必要があります。

5.5.3.2. ユーザーの初回署名証明書の取得

ユーザーの最初の署名証明書を承認する方法は 2 つあります。

- エージェントは CMC 要求を署名します。「エージェント証明書を使用した CMC 要求の署名」を参照してください。
- 証明書の登録は、共有シークレットを使用して認証されます。「共有シークレットを使用した証明書の登録の認証」 を参照してください。

5.5.3.2.1. エージェント証明書を使用した CMC 要求の署名

エージェント証明書を使用して CMC 要求に署名するプロセスは、「システム証明書およびサーバー証明書の取得」 で説明されているシステム証明書およびサーバー証明書の場合と同じです。唯一の違いは、ユーザーが CSR を作成し、承認のために CA エージェントに送信することです。

5.5.3.2.2. 共有シークレットを使用した証明書の登録の認証

ユーザーが最初の署名証明書を取得したいが、エージェントが、「エージェント証明書を使用した CMC 要求の署名」 で説明されているように要求を承認できない場合は、共有トークンを使用できます。このトークンを使用すると、ユーザーは最初の署名証明書を取得できます。次に、この証明書を使用してユーザーの他の証明書に署名できます。

このシナリオでは、Shared Secret のメカニズムを使用して、ユーザーの最初の署名証明書を取得します。「CMC 登録プロセス」 とともに以下の情報を使用します。

1. ユーザーまたは CA 管理者として共有トークンを作成します。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントガイド』 の 『共有シークレットワークフロー』 セクションを参照してください。

   以下の点に留意してください。

   - ユーザーがトークンを作成した場合、ユーザーはトークンを CA 管理者に送信する必要があります。
   - CA 管理者がトークンを作成した場合、管理者はユーザーがトークンを生成するのに使用するパスワードを共有する必要があります。セキュアな方法でパスワードを送信します。

2. CA 管理者として、LDAP のユーザーエントリーに Shared Token を追加します。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 「証明書の登録用ユーザーエントリーへの CMC 共有シークレットの追加」 および 『CMC 共有シークレット機能の有効化』セクションを参照してください。

3. **CMCRequest** ユーティリティーに渡される設定ファイルで以下のパラメーターを使用します。

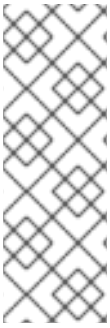
   - ***identification.enable***
   - ***witness.sharedSecret***
   - ***identityProofV2.enable***
   - ***identityProofV2.hashAlg***
   - ***identityProofV2.macAlg***
   - ***request.useSharedSecret***
   - ***request.privKeyId***

4. CA で必要な場合は、**CMCRequest** ユーティリティーに渡される設定ファイルで以下のパラメーターも使用します。

   - ***popLinkWitnessV2.enable***
   - ***popLinkWitnessV2.keyGenAlg***
   - ***popLinkWitnessV2.macAlg***

5.5.3.3. ユーザーの暗号化のみの証明書の取得

本セクションでは、既存のユーザー署名証明書で署名された暗号化のみの証明書を取得するワークフローを説明します。



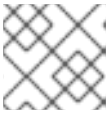
注記

ユーザーがさまざまな用途で複数の証明書を所有していて、1 つが署名している場合、ユーザーは最初に署名証明書を取得する必要があります。ユーザーが署名証明書を所有すると、CMC Shared Secret メカニズムを設定して依存することなく、Proof Of Origin に使用できます。

ユーザーの最初の署名証明書を取得する方法は、「ユーザーの初回署名証明書の取得」を参照してください。

ユーザーとして以下を行います。

1. Network Security Services (NSS) データベースまたはユーザーの署名証明書および鍵が含まれるスマートカードに保存されている暗号化トークンを使用します。
2. PKCS #10 形式または CRMF 形式で CSR を生成します。



注記

(キーのアーカイブが必要な場合は) CRMF 形式を使用してください。

3. CMC 要求を生成します。

   これは暗号のみの証明書であるため、秘密鍵は署名できません。そのため、Proof Of Possession (POP) は含まれていません。このため、登録には、2 つの手順が必要です。最初のリクエストが成功すると、**EncryptedPOP** 制御のある CMC 状態が生じます。次に、ユーザーは応答を使用して、**DecryptedPOP** 制御を含む CMC 要求を生成し、2 番目のステップで送信します。

   a. 最初のステップでは、デフォルトのパラメーターに加えて、ユーザーは、**CMCRequest** ユーティリティーに渡される設定ファイルに次のパラメーターを設定する必要があります。

      - ***identification.enable***
      - ***witness.sharedSecret***
      - ***identityProofV2.enable***
      - ***identityProofV2.hashAlg***
      - ***identityProofV2.macAlg***
      - ***popLinkWitnessV2.enable*** (CA で必要な場合)
      - ***popLinkWitnessV2.keyGenAlg*** (CA で必要な場合)
      - ***popLinkWitnessV2.macAlg*** (CA で必要な場合)
      - ***request.privKeyId***

      詳細は、CMCRequest(1) の man ページを参照してください。

      応答には以下が含まれます。

      - CMC で暗号化された POP コントロール

- **POP の required** エラーでの **CMCStatusInfoV2** コントロール
- リクエスト ID

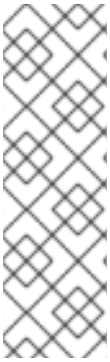
b. 次のステップでは、デフォルトのパラメーターに加えて、ユーザーは、**CMCRequest** ユーティリティーに渡される設定ファイルに次のパラメーターを設定する必要があります。

- ***decryptedPop.enable***
- ***encryptedPopResponseFile***
- ***decryptedPopRequestFile***
- ***request.privKeyId***

詳細は、CMCRequest(1) の man ページを参照してください。

5.5.3.3.1. キーアーカイブを使用した暗号化のみの証明書の取得例

キーアーカイブを使用して登録を実行するには、CRMF 要求にユーザーの暗号化された秘密鍵を含む CMC 要求を生成します。以下の手順は、ユーザーが署名証明書をすでに所有していることを前提としています。この署名証明書のニックネームは、手順の設定ファイルに設定されます。



注記

以下の手順は、署名に使用できない暗号のみの鍵で使用される 2 通の発行を説明します。証明書に署名できるキーを使用する場合は、**-q POP_NONE** の代わりに **-q POP_SUCCESS** オプションを、単一トリップ発行のために **CRMFPopClient** ユーティリティーに渡します。

POP_SUCCESS で **CRMFPoPClient** を使用する方法は、「**CRMFPopClient** を使用したキー Archival を持つ CSR の作成」 および 「**CRMFPopClient** を使用した SharedSecret ベースの CMC の CSR の作成」 を参照してください。

1. KRA トランスポート証明書を検索します。以下に例を示します。

```
$ pki cert-find --name KRA_transport_certificate_subject_CN
```

2. 前の手順で取得した KRA トランスポート証明書のシリアル番号を使用して、証明書をファイルに保存します。たとえば、**/home/user_name/kra.cert** ファイルに *12345* シリアル番号がある証明書を保存するには、次のコマンドを実行します。

```
$ pki cert-show 12345 --output /home/user_name/kra.cert
```

3. **CRMFPopClient** ユーティリティーを使用して以下を行います。

- キーアーカイブを使用して CSR を作成します。

1. 証明書が要求されるユーザーまたはエンティティーの証明書データベースディレクトリーに移動します。以下に例を示します。

```
$ cd /home/user_name/
```

2. RSA 秘密鍵が KRA トランスポート証明書によりラップされる CRMF 要求を作成するに

2. RSA 秘密鍵が KRA トランスポート証明書によりラップされる CRMF 要求を作成するには、**CRMFPopClient** ユーティリティーを使用します。たとえば、要求を **/home/user_name/crmf.req** ファイルに保存するには、以下のコマンドを実行します。

```
$ CRMFPopClient -d . -p token_password -n subject_DN -q POP_NONE \
  -b /home/user_name/kra.cert -w "AES/CBC/PKCS5Padding" \
  -v -o /home/user_name/crmf.req
```

コマンドで表示される秘密鍵の ID をメモします。ID は、2 番目のトリップの設定ファイルの ***request.privKeyId*** パラメーターの値として、後のステップで必要になります。

4. 以下の内容を含む、**/home/*user_name*/cmc.cfg** など、**CRMRequest** ユーティリティー用の設定ファイルを作成します。

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
input=/home/user_name/crmf.req

#output: full path for the CMC request in binary format
output=/home/user_name/cmc.req

#tokenname: name of token where agent signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for user certificate which will be used
#to sign the CMC full request.
nickname=signing_certificate

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.db which stores the agent certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf
```

5. CMC 要求を作成します。

```
$ CMCRequest /home/user_name/cmc.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

6. **/home/user_name/cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
#host: host name for the http server
host=server.example.com
```

```
#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in
#binary format
input=/home/user_name/cmc.req

#output: full path for the response in binary format
output=/home/user_name/cmc-response_round_1.bin

#tokenname: name of token where TLS client authentication cert can be found
#(default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/

#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert9.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=signing_certificate

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitUserSignedCMCFull?profileId=caFullCMCUserSignedCert
```

7. CMC 要求を CA に送信します。

```
$ HttpClient /home/user_name/cmc-submit.cfg
```

コマンドが成功すると、**HTTPClient** ユーティリティーは、CMC 応答を、設定ファイルの ***output*** パラメーターで指定されたファイルに保存します。

8. 応答ファイルを **CMCResponse** ユーティリティーに渡して応答を確認します。以下に例を示します。

```
$ CMCResponse -d /home/user_name/.dogtag/nssdb/ -i /home/user_name/cmc-
response_round_1.bin
```

最初のトリップが成功した場合は、**CMCResponse** は、以下のような出力を表示します。

```
Certificates:
  Certificate:
    Data:
```

```
      Version:  v3
      Serial Number: 0x1
      Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Issuer: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
      Validity:
        Not Before: Wednesday, May 17, 2017 6:06:50 PM PDT America/Los_Angeles
        Not  After: Sunday, May 17, 2037 6:06:50 PM PDT America/Los_Angeles
      Subject: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
...
Number of controls is 3
Control #0: CMC encrypted POP
  OID: {1 3 6 1 5 5 7 7 9}
   encryptedPOP decoded
Control #1: CMCStatusInfoV2
  OID: {1 3 6 1 5 5 7 7 25}
  BodyList: 1
  OtherInfo type: FAIL
   failInfo=POP required
Control #2: CMC ResponseInfo
  requestID: 15
```

9. 2 番目のトリップの場合は、後の手順で使用する **/home/user_name/cmc_DecryptedPOP.cfg** などの **DecryptedPOP** の設定ファイルを作成します。作成されたファイルに以下の内容を追加します。

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
#this field is actually unused in 2nd trip
input=/home/user_name/crmf.req

#output: full path for the CMC request in binary format
#this field is actually unused in 2nd trip
output=/home/user_name/cmc2.req

#tokenname: name of token where agent signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
nickname=signing_certificate

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.db which stores the agent
#certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf
```

```
decryptedPop.enable=true
encryptedPopResponseFile=/home/user_name/cmc-response_round_1.bin
request.privKeyId=-25aa0a8aad395ebac7e6a19c364f0dcb5350cfef
decryptedPopRequestFile=/home/user_name/cmc.DecryptedPOP.req
```

10. **DecryptPOP** CMC 要求を作成します。

```
$ CMCRequest /home/user_name/cmc.DecryptedPOP.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***decryptedPopRequestFile*** パラメーターで指定されたファイルに CMC 要求を保存します。

11. **/home/user_name/decrypted_POP_cmc-submit.cfg** などの **HttpClient** の設定ファイルを作成します。このファイルは、後で **DecryptedPOP** CMC 要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
#host: host name for the http server
host=server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in binary format
input=/home/user_name/cmc.DecryptedPOP.req

#output: full path for the response in binary format
output=/home/user_name/cmc-response_round_2.bin

#tokenname: name of token where TLS client authentication cert can be found (default is
internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/

#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert9.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=singing_certificate

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitUserSignedCMCFull?profileId=caFullCMCUserCert
```

12. **DecryptedPOP** CMC 要求を CA に送信します。

```
$ HttpClient /home/user_name/decrypted_POP_cmc-submit.cfg
```

コマンドが成功すると、**HTTPClient** ユーティリティーは、CMC 応答を、設定ファイルの ***output*** パラメーターで指定されたファイルに保存します。

13. CMC の応答を PKCS #7 証明書チェーンに変換するには、**CMCResponse** ユーティリティーの ***-i*** パラメーターに CMC レスポンスファイルを渡します。以下に例を示します。

```
$ CMCResponse -i /home/user_name/cmc-response_round_2.bin -o
/home/user_name/certs.p7
```

または、個々の証明書を PEM 形式で表示するには、**-v** ユーティリティーに渡します。

次のトリップが成功した場合は、**CMCResponse** は、以下のような出力を表示します。

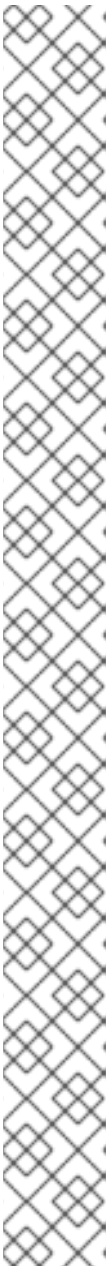
```
Certificates:
  Certificate:
    Data:
      Version:  v3
      Serial Number: 0x2D
      Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
      Issuer: CN=CA Signing Certificate,OU=pki-tomcat,O=unknown00262DFC6A5E Security
Domain
      Validity:
        Not Before: Thursday, June 15, 2017 3:43:45 PM PDT America/Los_Angeles
        Not  After: Tuesday, December 12, 2017 3:43:45 PM PST America/Los_Angeles
      Subject: CN=user_name,UID=example,OU=keyArchivalExample
...
Number of controls is 1
Control #0: CMCStatusInfo
  OID: {1 3 6 1 5 5 7 7 1}
  BodyList: 1
  Status: SUCCESS
```

5.6. 一括発行の実行

管理者が多数の証明書を同時に送信および生成する必要がある場合があります。Certificate System で提供されるツールの組み合わせを使用して、証明書要求を含むファイルを CA に送信できます。この手順例では、リクエストを生成する **PKCS10Client** コマンドと、CA に要求を送信する **sslget** コマンドを使用します。

1. このプロセスはスクリプト化されているため、CA (ホスト、ポート) と、認証に使用されるアイテム (エージェント証明書と証明書データベースおよびパスワード) を識別するために複数の変数を設定する必要があります。たとえば、ターミナルでエクスポートして、セッションに以下の変数を設定します。

```
export d=/var/tmp/testDir
export p=password
export f=/var/tmp/server.csr.txt
export nick="CA agent cert"
export cahost=1.2.3.4
export caport=8443
```



注記

ローカルシステムには、エージェントの証明書を持つ有効なセキュリティーデータベースが必要です。データベースを設定するには、以下を行います。

a. ブラウザーからエージェントユーザー証明書および鍵をエクスポートまたはダウンロードし、**agent.p12** などのファイルに保存します。

b. 必要に応じて、セキュリティーデータベース用の新しいディレクトリーを作成します。

```
mkdir ${d}
```

c. 必要な場合は、新規セキュリティーデータベースを作成します。

```
certutil -N -d ${d}
```

d. Certificate System インスタンスを停止します。

```
pki-server stop instance_name
```

e. **pk12util** を使用して証明書をインポートします。

```
# pk12util -i /tmp/agent.p12 -d ${d} -W p12filepassword
```

手順に成功すると、コマンドは以下の出力を出力します。

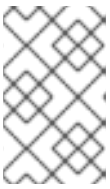
```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

f. Certificate System インスタンスを起動します。

```
pki-server start instance_name
```

2. 2 つの追加の変数を設定する必要があります。要求の処理に使用される CA プロファイルを識別する変数、およびプロファイルフォームの情報を提供するための post ステートメントの送信に使用される変数。

```
export
post="cert_request_type=pkcs10&xmlOutput=true&profileId=caAgentServerCert&cert_request=
"
export url="/ca/ee/ca/profileSubmitSSLClient"
```



注記

この例では、証明書要求を **caAgentServerCert** プロファイルに送信します (ただし、**post** ステートメントの **profileId** 要素で識別されます)。カスタムプロファイルを含む任意の証明書プロファイルを使用できます。

3. 変数設定をテストします。

```
echo ${d} ${p} ${f} ${nick} ${cahost} ${caport} ${post} ${url}
```

4. (この例では) **PKCS10Client** を使用して証明書要求を生成します。

```
time for i in {1..10}; do /usr/bin/PKCS10Client -d ${d} -p ${p} -o ${f}.${i} -s
"cn=testms${i}.example.com"; cat ${f}.${i} >> ${f}; done

perl -pi -e 's/\r\n//;s/\+/%2B/g;s/\//%2F/g' ${f}

wc -l ${f}
```

5. 手順 4 で作成した一括証明書要求ファイルを、**sslget** を使用する CA プロファイルインターフェイスに送信します。以下に例を示します。

```
cat ${f} | while read thisreq; do /usr/bin/sslget -n "${nick}" -p ${p} -d ${d} -e ${post}${thisreq} -
v -r ${url} ${cahost}:${caport}; done
```

5.7. CISCO ルーターでの証明書の登録

Cisco によって設計された Simple Certificate Enrollment Protocol (SCEP) は、ルーターが CA などの証明書発行機関と通信して、ルーターの証明書を登録するための方法です。

通常、ルーターインストーラーは CA の URL とチャレンジパスワード (ワンタイム PIN とも呼ばれます) をルーターに入力し、コマンドを発行して登録を開始します。次に、ルーターは SCEP を介して CA と通信し、証明書を生成、要求、および取得します。ルーターは、SCEP を使用して保留中の要求のステータスを確認することもできます。

5.7.1. SCEP 登録の有効化

セキュリティー上の理由から、SCEP 登録は CA でデフォルトで無効になっています。ルーターの登録を可能にするには、CA に対して SCEP 登録を手動で有効にする必要があります。

1. 設定ファイルを編集できるように CA サーバーを停止します。

```
pki-server stop instance_name
```

2. CA の **CS.cfg** ファイルを開きます。

```
vim /var/lib/pki/instance_name/ca/conf/CS.cfg
```

3. ***ca.scep.enable*** を true に設定します。パラメーターが存在しない場合は、パラメーターで行を追加します。

```
ca.scep.enable=true
```

4. CA サーバーを起動します。

```
pki-server start instance_name
```

5.7.2. SCEP のセキュリティー設定の設定

管理者は、登録認証と通常の証明書登録に同じ証明書を使用しない、または接続強度の低下を防ぐために許可された暗号化アルゴリズムを設定するなど、いくつかの異なるパラメーターを使用して、SCEP 接続に特定のセキュリティー要件を設定できます。これらのパラメーターを 表5.1「SCEP セキュリ

ティーの設定パラメーター」 に記載します。

表5.1 SCEP セキュリティーの設定パラメーター

パラメーター	説明
ca.scep.encryptionAlgorithm	デフォルトまたは優先暗号化アルゴリズムを設定します。
ca.scep.allowedEncryptionAlgorithms	許可される暗号化アルゴリズムのコンマ区切りリストを設定します。
ca.scep.hashAlgorithm	デフォルトまたは優先ハッシュアルゴリズムを設定します。
ca.scep.allowedHashAlgorithms	許可されるハッシュアルゴリズムのコンマ区切りリストを設定します。
ca.scep.nickname	SCEP 通信に使用する証明書のニックネームを指定します。このパラメーターが設定されていない限り、デフォルトで CA のキーペアおよび証明書が使用されます。
ca.scep.nonceSizeLimit	SCEP リクエストに許可される最大 nonce サイズ (バイト単位) を設定します。デフォルトは 16 バイトです。

SCEP 登録の接続にセキュリティー設定を設定するには、以下を実行します。

1. 設定ファイルを編集できるように CA サーバーを停止します。

```
pki-server stop instance_name
```

2. CA の **CS.cfg** ファイルを開きます。

```
vim /var/lib/pki/instance_name/ca/conf/CS.cfg
```

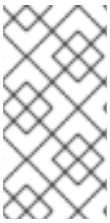
3. 表5.1「SCEP セキュリティーの設定パラメーター」 に記載されているように、必要なセキュリティーパラメーターを設定します。このパラメーターが存在しない場合は、**CS.cfg** ファイルに追加します。

```
ca.scep.encryptionAlgorithm=DES3
ca.scep.allowedEncryptionAlgorithms=DES3
ca.scep.hashAlgorithm=SHA1
ca.scep.allowedHashAlgorithms=SHA1,SHA256,SHA512
ca.scep.nickname=Server-Cert
ca.scep.nonceSizeLimit=20
```

4. CA サーバーを起動します。

```
pki-server start instance_name
```

5.7.3. SCEP 登録のルーターの設定



注記

ルーター IOS の全バージョンには関連する暗号化機能があるわけではありません。ファームウェアイメージに認証局の相互運用性があることを確認します。証明書システム SCEP サポートは、IOS C2600 Software (C2600-JK9S-M), バージョン 12.2(40), RELEASE SOFTWARE (fc1) を実行している Cisco 2611 ルーターでテストされました。

ルーターに SCEP 証明書を登録する前に、ルーターが適切に設定されていることを確認します。

- ルーターは、IP アドレス、DNS サーバー、およびルーティング情報で設定する必要があります。
- ルーターの日付/時刻が正しく設定されている必要があります。
- ルーターのホスト名と dnsname を設定する必要があります。

ルーターのハードウェアの設定方法は、ルーターのドキュメントを参照してください。

5.7.4. ルーターの SCEP 証明書の生成

以下の手順では、ルーターの SCEP 証明書を生成する方法を説明します。

1. ランダムな PIN を選択します。
2. ルーターが CA に対して直接認証できるように、PIN とルーターの ID を **flatfile.txt** ファイルに追加します。以下に例を示します。

   ```
   vim /var/lib/pki/instance_name/ca/conf/flatfile.txt

   UID:172.16.24.238
   PWD:Uojs93wkfd0IS
   ```

   PWD 行の後に空の行を挿入してください。

   ルーターの IP アドレスは、IPv4 アドレスまたは IPv6 アドレスになります。

   フラットファイルの認証の使用は、「フラットファイル認証の設定」 に記載されています。
3. ルーターのコンソールにログインします。以下の例では、ルーターの名前は **scep** です。

   ```
   scep>
   ```
4. 特権コマンドを有効にします。

   ```
   scep> enable
   ```
5. 設定モードを入力します。

   ```
   scep# conf t
   ```

6. root で始まり、証明書チェーン内のすべての CA に CA 証明書をインポートします。たとえば、次のコマンドシーケンスは、チェーン内の 2 つの CA 証明書をルーターにインポートします。

```
scep(config)# crypto ca trusted-root1
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 1
scep(config)# crypto ca trusted-root0
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 0
```

7. CA アイデンティティーを設定し、SCEP 登録プロファイルにアクセスするための URL を入力します。CA の場合の例を以下に示します。

```
scep(config)# crypto ca identity CA
scep(ca-identity)# enrollment url http://server.example.com:8080/ca/cgi-bin
scep(ca-identity)# crl optional
```

8. CA の証明書を取得します。

```
scep(config)# crypto ca authenticate CA
Certificate has the following attributes:
Fingerprint: 145E3825 31998BA7 F001EA9A B4001F57
% Do you accept this certificate? [yes/no]: yes
```

9. RSA 鍵ペアを生成します。

```
scep(config)# crypto key generate rsa
The name for the keys will be: scep.server.example.com
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.

How many bits in the modulus [512]:
Generating RSA keys ...
[OK]
```

10. 最後に、ルーターに証明書を生成します。

```
scep(config)# crypto ca enroll CA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
password to the CA Administrator in order to revoke your certificate.
For security reasons your password will not be saved in the configuration.
Please make a note of it.

Password: secret
Re-enter password: secret
```

```
% The subject name in the certificate will be: scep.server.example.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 57DE391C
% Include an IP address in the subject name? [yes/no]: yes
% Interface: Ethernet0/0
% Request certificate from CA? [yes/no]: yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

% Fingerprint:D89DB555 E64CC2F7 123725B4 3DBDF263

Jan 12 13:41:17.348: %CRYPTO-6-CERTRET: Certificate received from Certificate
```

11. 設定モードを閉じます。

```
scep(config)# exit
```

12. ルーターが適切に登録されたことを確認するには、ルーターに保存されている証明書の一覧を表示します。

```
scep# show crypto ca certificates
Certificate
 Status: Available
 Certificate Serial Number: 0C
 Key Usage: General Purpose
 Issuer:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Subject Name Contains:
 Name: scep.server.example.com
 IP Address: 10.14.1.94
 Serial Number: 57DE391C
 Validity Date:
 start date: 21:42:40 UTC Jan 12 2007
 end date: 21:49:50 UTC Dec 31 2008
 Associated Identity: CA

CA Certificate
 Status: Available
 Certificate Serial Number: 01
 Key Usage: Signature
 Issuer:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Subject:
 CN = Certificate Authority
  O = Sfbay Red hat Domain 20070111d12
 Validity Date:
 start date: 21:49:50 UTC Jan 11 2007
 end date: 21:49:50 UTC Dec 31 2008
 Associated Identity: CA
```

5.7.5. Subordinate CA の使用

ルーターが CA に対して認証できるようにするには、ルートから CA 証明書チェーンのすべての CA 証明書をルーターにインポートする必要があります。たとえば、次のコマンドシーケンスは、チェーン内の 2 つの CA 証明書をルーターにインポートします。

```
scep(config)# crypto ca trusted-root1
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 1
scep(config)# crypto ca trusted-root0
scep(ca-root)# root CEP http://server.example.com:8080/ca/cgi-bin/pkiclient.exe
scep(ca-root)# crl optional
scep(ca-root)# exit
scep(config)# cry ca authenticate 0
```

CA 証明書に CRL ディストリビューションポイントの拡張が設定されていない場合は、**optional** に設定して CRL 要件をオフにします。

```
scep(ca-root)# crl optional
```

その後、「ルーターの SCEP 証明書の生成」 の説明に従って CA アイデンティティーを設定します。

5.7.6. ルーターの再登録

ルーターを新しい証明書で再登録できるようにするには、既存の設定を削除する必要があります。

1. 既存のキーを削除 (ゼロ化)。

```
scep(config)# crypto key zeroize rsa
% Keys to be removed are named scep.server.example.com.
Do you really want to remove these keys? [yes/no]: yes
```

2. CA アイデンティティーを削除します。

```
scep(config)# no crypto ca identity CA
% Removing an identity will destroy all certificates received from
the related Certificate Authority.

Are you sure you want to do this? [yes/no]: yes
% Be sure to ask the CA administrator to revoke your certificates.

No enrollment sessions are currently active.
```

5.7.7. デバッグの有効化

ルーターは、debug ステートメントを有効にして、SCEP 操作中に追加のデバッグを提供します。

```
scep# debug crypto pki callbacks
Crypto PKI callbacks debugging is on

scep# debug crypto pki messages
Crypto PKI Msg debugging is on
```

```
scep# debug crypto pki transactions
Crypto PKI Trans debugging is on

scep#debug crypto verbose
verbose debug output debugging is on
```

5.7.8. SCEP で ECC 証明書を発行

デフォルトでは、ECCCA はすぐに使用できる SCEP をサポートしていません。ただし、指定した RSA 証明書を使用して、以下の 2 つの領域を処理することで回避できます。

- 暗号化/複号証明書 - 暗号化機能/複号機能を持つ RSA 証明書 (以下の例では scepRSAcert) を指定します。
- 署名証明書 - 自己署名ではなく、クライアント側で使用する RSA 証明書を取得します (以下の例では signingCert 証明書)。

たとえば、scepRSAcert 証明書が暗号化/複号証明書で、署名証明書である SignCert を使用する場合は、次のコマンドを実行します。

```
sscep enroll -c ca.crt -e scepRSAcert.crt -k local.key -r local.csr -K sign.key -O sign.crt -E 3des -S
sha256 -l cert.crt -u 'http://example.example.com:8080/ca/cgi-bin/pkiclient.exe'
```

5.8. 証明書の透過性の使用

Certificate System は、証明書 Certificate Transparency (CT) V1 サポートの基本バージョン (rfc 6962) を提供します。各デプロイメントサイトがルート CA 証明書を含めることを選択した信頼できるログから、Signed Certificate Time スタンプ (SCT) が埋め込まれた証明書を発行する機能があります。また、複数の CT ログに対応するようにシステムを設定することもできます。この機能を使用するには、少なくとも 1 つの信頼できる CT ログが必要です。

重要

デプロイメントサイトが、信頼できる CT ログサーバーとの信頼関係を確立します。

証明書の透過性設定を設定する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『証明書の透過性のテスト』セクションを参照してください。

5.8.1. 証明書の透過性のテスト

CT 設定のテスト方法の例として、以下の手順では Google CT テストログに対する実際のテストを説明します。より包括的なテスト手順では、TLS サーバーを設定し、指定された CT ログからその証明書が含まれるかどうかをテストします。ただし、次のコマンドは、証明書が発行された後に SCT 拡張を含むことを確認するクイックテストとして機能します。

テスト手順は、**openssl** を使用して SCT 拡張を検証するために、Certificate Signing Request (CSR) を生成して送信することで設定されます。**CS.cfg** ファイルのテスト設定は次のとおりです。

```
ca.certTransparency.mode=enabled
ca.certTransparency.log.1.enable=true
ca.certTransparency.log.1.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEw8i8S7qiGEs9NXv
0ZJFh6uuOm<snip>
ca.certTransparency.log.1.url=http://ct.googleapis.com:80/testtube/
```

```
ca.certTransparency.log.1.version=1
ca.certTransparency.log.2.enable=true
ca.certTransparency.log.2.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKATl2B3SAbxyzG
OfNRB+AytNTG<snip>
ca.certTransparency.log.2.url=http://ct.googleapis.com:80/logs/crucible/
ca.certTransparency.log.2.version=1
ca.certTransparency.log.3.enable=false
ca.certTransparency.log.3.pubKey=MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEiKfWtuoWCPMEzS
KySjMjXpo38W<snip>
ca.certTransparency.log.3.url=http://ct.googleapis.com:80/logs/solera2020/
ca.certTransparency.log.3.version=1
ca.certTransparency.log.num=3
```

1. まず、CSR を生成します。以下に例を示します。

```
# PKCS10Client -d . -p passwd -l 2048 -n "cn=user.test.domain.com,OU=user-
TEST,O=TestDomain" -o pkcs10-TLS.req
```

2. 次に、**CS.cfg** の **ca.certTransparency.mode** パラメーターで定義される CT モードに応じて CSR を登録プロファイルに送信します。

   - パラメーターが **有効** な場合は、登録プロファイルを使用します。
   - このパラメーターが **perProfile** に設定されている場合には、CT プロファイルのいずれかを使用します (例: **caServerCertWithSCT**)。

3. 発行した b64 証明書をファイルにコピーします (例: **.ct1.pem**)。

4. pem をバイナリーに変換します。

```
#  AtoB ct1.pem ct1.bin
```

5. DER 証明書の内容を表示します。

```
#  openssl x509 -noout -text -inform der -in ct1.bin
```

6. SCT 拡張が存在することを確認します。以下に例を示します。

```
        CT Precertificate SCTs:
         Signed Certificate Timestamp:
           Version   : v1 (0x0)
           Log ID    : B0:CC:83:E5:A5:F9:7D:6B:AF:7C:09:CC:28:49:04:87:
                       2A:C7:E8:8B:13:2C:63:50:B7:C6:FD:26:E1:6C:6C:77
           Timestamp : Jun 11 23:07:14.146 2020 GMT
           Extensions: none
           Signature : ecdsa-with-SHA256
                       30:44:02:20:6E:E7:DC:D6:6B:A6:43:E3:BB:8E:1D:28:
                       63:C6:6B:03:43:4E:7A:90:0F:D6:2B:E8:ED:55:1D:5F:
                       86:0C:5A:CE:02:20:53:EB:75:FA:75:54:9C:9F:D3:7A:
                       D4:E7:C6:6C:9B:33:2A:75:D8:AB:DE:7D:B9:FA:2B:19:
                       56:22:BB:EF:19:AD
         Signed Certificate Timestamp:
           Version   : v1 (0x0)
           Log ID    : C3:BF:03:A7:E1:CA:88:41:C6:07:BA:E3:FF:42:70:FC:
```

```
            A5:EC:45:B1:86:EB:BE:4E:2C:F3:FC:77:86:30:F5:F6
        Timestamp : Jun 11 23:07:14.516 2020 GMT
        Extensions: none
        Signature : ecdsa-with-SHA256
            30:44:02:20:4A:C9:4D:EF:64:02:A7:69:FF:34:4E:41:
            F4:87:E1:6D:67:B9:07:14:E6:01:47:C2:0A:72:88:7A:
            A9:C3:9C:90:02:20:31:26:15:75:60:1E:E2:C0:A3:C2:
            ED:CF:22:A0:3B:A4:10:86:D1:C1:A3:7F:68:CC:1A:DD:
            6A:5E:10:B2:F1:8F
```

また、asn1 ダンプを実行して SCT を確認します。

```
# openssl asn1parse -i -inform der -in ct1.bin
```

また、以下のように 16 進ダンプを確認します。

```
  740:d=4  hl=4 l= 258 cons:     SEQUENCE
  744:d=5  hl=2 l=  10 prim:      OBJECT            :CT Precertificate SCTs
  756:d=5  hl=3 l= 243 prim:      OCTET STRING      [HEX
DUMP]:0481F000EE007500B0CC83E5A5F97D6B<snip>
```

第6章 TOKEN MANAGEMENT SYSTEM の使用および設定: TPS および TKS

本章では、**HSM** または **トークン** と呼ばれるハードウェアセキュリティーモジュールを使用して、Certificate System インスタンスの証明書および鍵を生成および保存する手順を説明します。

本章には管理手順のみが含まれています。Token Management System の概念に関する一般的な情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』を参照してください。

6.1. TPS プロファイル

注記

一般情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『TPS プロファイル』セクションを参照してください。

個々のファイルまたは LDAP で定義および保存される CA 登録プロファイルとは異なり、TPS プロファイル (トークンタイプとも呼ばれます) は TPS 設定ファイル **CS.cfg** で定義されます。

TPS プロファイル (トークンタイプ) の設定パラメーターは、以下の形式で設定されます。

```
op.<explicit op>.<profile id>.<implicit op>.<key type>.*
```

上記の *<explicit op>* および *<implicit op>* は、以下の TPS 操作セクションで説明される明示的な操作および暗黙的な操作のいずれかであり、*<key type>* は各証明書タイプに指定された名前になります。

設定パラメーターの例は、次の例のようになります。

```
op.enroll.userKey.keyGen.encryption.*
```

6.2. TPS 操作

明示的な操作

明示的な操作 はユーザーが呼び出す操作です。明示的な操作には、**enroll** (**op.enroll.***)、**format** (**op.format.***)、および **pinReset** (**op.pinReset.***) が含まれます。

暗黙的な操作

暗黙的な操作 は、明示的な操作が処理されるときにトークンのポリシーまたはステータスが原因で発生する操作です。暗黙的な操作には、**keyGen** (**op.enroll.userKey.keyGen.***)、**renewal** (**op.enroll.userKey.renewal.***)、**update.applet** (**op.enroll.userKey.update.applet.***)、キー更新 (**op.enroll.userKey.update.symmetricKeys.***) が含まれます。

暗黙的な操作の一部は、キーのタイプごとに制御されます。これには、**recovery**、**serverKeygen**、および **revocation** が含まれます。

次の TPS プロファイルの例では、サーバー側で生成されるユーザーキーを指定しています。

```
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true
```

さらに、次の例は、状態遷移中、キーが侵害されたトークンが失効理由 **1** で認証を取り消す必要があることを TPS に通知します。

```
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert.reason=1
```

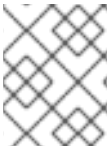
RFC 5280 に基づいて、失効可能な理由とそれらのコードは以下のように定義されます。

表6.1 失効理由およびコード

理由	コード
指定なし	0
keyCompromise	1
CACompromise	2
affiliationChanged	3
superseded	4
cessationOfOperation	5
certificateHold	6
removeFromCRL	8
privilegeWithdrawn	9
AACompromise	10

6.3. トークンポリシー

本セクションでは、TPS UI を使用して、トークンごとに適用可能なトークンポリシーの一覧を提供します。各セクションでは、各ポリシーが設定にどのように反映されるかを示します。



注記

一般情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントガイド』の『トークンポリシー』セクションを参照してください。

ポリシーは、セミコロン (";"") で区切られたポリシーの集合体です。各ポリシーは、キーワード **YES** または **NO** を使用してオンまたはオフにすることができます。以下のリストの各ポリシーは、デフォルト値 (設定がポリシー文字列にまったく存在しなかった場合に TPS によって実行されるアクション) で導入されます。

RE_ENROLL=YES

このポリシーは、トークンが再登録操作を許可するかどうかを制御します。これにより、すでに登録されたトークン (証明書を含む) を再登録し、新しいトークンを登録できるようになります。これを **NO** に設定すると、再登録を試行するとサーバーはエラーを返します。

このポリシーでは、特別な設定は必要ありません。登録は、標準の登録プロファイルで続行されます。このプロファイルは、最初にトークンを登録する可能性があります。

RENEW=NO;RENEW_KEEP_OLD_ENC_CERTS=YES

更新により、トークンは、プロファイルで生成された証明書をトークンの所定の場所で更新することができます。**RENEW** を **YES** に設定すると、Enterprise Security Client (ESC) からの簡単な登録により、上記のように再登録せずに更新が行われます。

RENEW_KEEP_OLD_ENC_CERTS 設定は、更新操作が以前のバージョンの暗号化証明書を保持するかどうかを決定します。以前の証明書を保持すると、ユーザーは古い証明書で暗号化されたデータにアクセスできます。このオプションを **NO** に設定すると、古い証明書で暗号化されたものはすべて復元できなくなります。

設定:

```
op.enroll.userKey.renewal.encryption.ca.conn=ca1
op.enroll.userKey.renewal.encryption.ca.profileId=caTokenUserEncryptionKeyRenewal
op.enroll.userKey.renewal.encryption.certAttrId=c2
op.enroll.userKey.renewal.encryption.certId=C2
op.enroll.userKey.renewal.encryption.enable=true
op.enroll.userKey.renewal.encryption.gracePeriod.after=30
op.enroll.userKey.renewal.encryption.gracePeriod.before=30
op.enroll.userKey.renewal.encryption.gracePeriod.enable=false
op.enroll.userKey.renewal.keyType.num=2
op.enroll.userKey.renewal.keyType.value.0=signing
op.enroll.userKey.renewal.keyType.value.1=encryption
op.enroll.userKey.renewal.signing.ca.conn=ca1
op.enroll.userKey.renewal.signing.ca.profileId=caTokenUserSigningKeyRenewal
op.enroll.userKey.renewal.signing.certAttrId=c1
op.enroll.userKey.renewal.signing.certId=C1
op.enroll.userKey.renewal.signing.enable=true
op.enroll.userKey.renewal.signing.gracePeriod.after=30
op.enroll.userKey.renewal.signing.gracePeriod.before=30
op.enroll.userKey.renewal.signing.gracePeriod.enable=false
```

このタイプの更新設定は、更新固有の追加設定をいくつか追加し、基本的な **userKey** 標準登録プロファイルをミラーリングします。このパリティーが必要なのは、更新が開始される前に、トークンに最初に登録された証明書の数とタイプを正確に更新するためです。

FORCE_FORMAT=NO

このポリシーにより、登録操作ごとにフォーマット操作が要求されます (有効な場合)。これは、ユーザーが管理者に返すことなくトークンをリセットできるようにする最終手順です。これを **YES** に設定すると、ユーザーが開始された登録操作がすべて形式になり、トークンがフォーマットされた状態に対して強制的にリセットされます。

追加の設定は必要ありません。単純な形式は、標準のフォーマット操作の実行に使用されるものと同じ TPS プロファイルで実行されます。

PIN_RESET=NO

このポリシーは、すでに登録されているトークンが ESC を使用して明示的なピンリセット変更を実行できるかどうかを決定します。この値は、**YES** に設定しなければならないか、サーバーがエラーにより発生した操作は拒否されます。

設定:

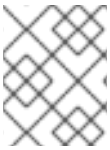
```
op.enroll.userKey.pinReset.enable=true
op.enroll.userKey.pinReset.pin.maxLen=10
op.enroll.userKey.pinReset.pin.maxRetries=127
op.enroll.userKey.pinReset.pin.minLen=4
```

上記の例では、**minLen** および **maxLen** の設定が、選択したパスワードの長さに制約を課し、**maxRetries** 設定は、ロックアップする前に指定された回数の再試行のみを許可するようにトークンを設定します。

TPS ポリシーは、最新の TPS ユーザーインターフェイスを使用して簡単に編集できます。ポリシーの変更を必要とするトークンに移動し、**Edit** をクリックします。これにより、フィールドを編集できるダイアログが表示されます。これは、セミコロンで区切られたポリシーのコレクションです。サポートされている各ポリシーは、TPS によって認識されるように ***<POLICYNAME>*=YES** または ***<POLICYNAME>*=NO** に設定する必要があります。

6.4. トークン操作およびポリシー処理

このセクションでは、トークンに関連する主要な操作 (明示的および暗黙的) について説明します。以下のリストは、各機能とその設定について記載しています。



注記

一般情報は、『Red Hat Certificate System 計画、インストール、デプロイメントのガイド』 の 『トークンポリシー』 セクションを参照してください。

形式

(ユーザーが開始した) Format 操作は、製造元から提供された完全に空白の状態のトークンを受け取り、Coolkey アプレットを読み込みます。

設定例:

```
#specify that we want authentication for format. We almost always want this at true:
op.format.userKey.auth.enable=true
#specify the ldap authentication configuration, so TPS knows where to validate credentials:
op.format.userKey.auth.id=ldap1
#specify the connection the the CA
op.format.userKey.ca.conn=ca1
#specify id of the card manager applet on given token
op.format.userKey.cardmgr_instance=A0000000030000

#specify if we need to match the visa cuid to the nist sp800sp derivation algorithm KDD value.
Mostly will be false:
op.format.userKey.cuidMustMatchKDD=false

#enable ability to restrict key changoever to a specific range of key set:
op.format.userKey.enableBoundedGPKeyVersion=true
#enable the phone home url to write to the token:
```

```
op.format.userKey.issuerinfo.enable=true
#actual home url to write to token:
op.format.userKey.issuerinfo.value=http://server.example.com:8080/tps/phoneHome
#specify whether to request a login from the client. Mostly true, external reg may want this to be
false:
op.format.userKey.loginRequest.enable=true
#Actual range of desired keyset numbers:
op.format.userKey.maximumGPKeyVersion=FF
op.format.userKey.minimumGPKeyVersion=01
#Whether or not to revoke certs on the token after a format, and what the reason will be if so:
op.format.userKey.revokeCert=true
op.format.userKey.revokeCert.reason=0
#This will roll back the reflected keyyset version of the token in the tokendb. After a failed key
changeover operation. This is to keep the value in sync with reality in the tokendb. Always false,
since this version of TPS avoids this situation now:
op.format.userKey.rollbackKeyVersionOnPutKeyFailure=false

#specify connection to the TKS:
op.format.userKey.tks.conn=tks1
#where to get the actual applet file to write to the token:
op.format.userKey.update.applet.directory=/usr/share/pki/tps/applets
#Allows a completely blank token to be recognized by TPS. Mostly should be true:
op.format.userKey.update.applet.emptyToken.enable=true
#Always should be true, not supported:
op.format.userKey.update.applet.encryption=true
#Actual version of the applet file we want to upgrade to. This file will have a name something like:
1.4.54de7a99.ijc:
op.format.userKey.update.applet.requiredVersion=1.4.54de790f
#Symm key changeover:
op.format.userKey.update.symmetricKeys.enable=false
op.format.userKey.update.symmetricKeys.requiredVersion=1
#Make sure the token db is in sync with reality. Should always be true:
op.format.userKey.validateCardKeyInfoAgainstTokenDB=true
```

登録

基本的な登録操作では、フォーマットされたトークンを取得し、トークンをカスタマイズするために証明書とキーをトークンに配置します。次の設定例では、これを制御する方法を説明します。

この例は、更新および内部リカバリーを処理しない基本的な登録を示しています。ここで説明されていない設定は、Format セクションで説明されています。または必須ではありません。

```
op.enroll.userKey.auth.enable=true
op.enroll.userKey.auth.id=ldap1
op.enroll.userKey.cardmgr_instance=A0000000030000
op.enroll.userKey.cuidMustMatchKDD=false

op.enroll.userKey.enableBoundedGPKeyVersion=true
op.enroll.userKey.issuerinfo.enable=true
op.enroll.userKey.issuerinfo.value=http://server.example.com:8080/tps/phoneHome

#configure the encryption cert and keys  we want on the token:

#connection the the CA, which issues the certs:
op.enroll.userKey.keyGen.encryption.ca.conn=ca1
#Profile id we want the CA to use to issue our encrytion cert:
```

```
op.enroll.userKey.keyGen.encryption.ca.profileId=caTokenUserEncryptionKeyEnrollment

#These two cover the indexes of the certs written to the token. Each cert needs a unique index or
“slot”. In our sample the enc cert will occupy slot 2 and the signing cert, shown later, will occupy
slot 1. Avoid overlap with these numbers:
op.enroll.userKey.keyGen.encryption.certAttrId=c2
op.enroll.userKey.keyGen.encryption.certId=C2

op.enroll.userKey.keyGen.encryption.cuid_label=$cuid$
#specify size of generated private key:
op.enroll.userKey.keyGen.encryption.keySize=1024
op.enroll.userKey.keyGen.encryption.keyUsage=0
op.enroll.userKey.keyGen.encryption.keyUser=0
#specify pattern for what the label of the cert will look like when the cert nickname is displayed in
browsers and mail clients:
op.enroll.userKey.keyGen.encryption.label=encryption key for $userid$
#specify if we want to overwrite certs on a re-enrollment operation. This is almost always the case:
op.enroll.userKey.keyGen.encryption.overwrite=true

#The next several settings specify the capabilities that the private key on the final token will inherit.
For instance this will determine if the cert can be used for encryption or digital signatures. There
are settings for both the private and public key.

op.enroll.userKey.keyGen.encryption.private.keyCapabilities.decrypt=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.derive=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.encrypt=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.private=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.sensitive=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.sign=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.token=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.unwrap=true
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.verify=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.encryption.private.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.encryption.privateKeyAttrId=k4
op.enroll.userKey.keyGen.encryption.privateKeyNumber=4
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.derive=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.encrypt=true
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.private=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.sensitive=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.sign=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.token=true
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.verify=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.encryption.public.keyCapabilities.wrap=true

#The following index numbers correspond to the index or slot that the private and public keys
occupy. The common formula we use is that the public key index will be 2 * cert id + 1, and the
private key index, shown above will be 2 * cert id. In this example the cert id is 2, so the key ids will
be 4 and 5 respectively. When composing these, be careful not to create conflicts. This applies to
the signing key section below.
```

```
op.enroll.userKey.keyGen.encryption.publicKeyAttrId=k5
op.enroll.userKey.keyGen.encryption.publicKeyNumber=5

#specify if, when a certificate is slated for revocation, based on other rules, we want to check to see if some other token is using this cert in a shared situation. If this is set to true, and this situation is found the cert will not be revoked until the last token wants to revoke this cert:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.holdRevocationUntilLastCredential=false


#specify, if we want server side keygen, if we want to have that generated key archived to the drm. This is almost always the case, since we want the ability to later recover a cert and its encryption private key back to a new token:
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
#connection to drm to generate the key for us:
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
#specify server side keygen of the encryption private key. This most often will be desired:
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true

#This setting tells us how many certs we want to enroll for this TPS profile, in the case “userKey”. Here we want 2 total certs. The next values then go on to index into the config what two types of certs we want, signing and encryption:
op.enroll.userKey.keyGen.keyType.num=2
op.enroll.userKey.keyGen.keyType.value.0=signing
op.enroll.userKey.keyGen.keyType.value.1=encryption

#configure the signing cert and keys we want on the token the settings for these are similar to the encryption settings already discussed, except the capability flags presented below, since this is a signing key.

op.enroll.userKey.keyGen.signing.ca.conn=ca1
op.enroll.userKey.keyGen.signing.ca.profileId=caTokenUserSigningKeyEnrollment
op.enroll.userKey.keyGen.signing.certAttrId=c1
op.enroll.userKey.keyGen.signing.certId=C1
op.enroll.userKey.keyGen.signing.cuid_label=$cuid$
op.enroll.userKey.keyGen.signing.keySize=1024
op.enroll.userKey.keyGen.signing.keyUsage=0
op.enroll.userKey.keyGen.signing.keyUser=0
op.enroll.userKey.keyGen.signing.label=signing key for $userid$
op.enroll.userKey.keyGen.signing.overwrite=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.derive=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.encrypt=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.private=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.sensitive=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.sign=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.signRecover=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.token=true
op.enroll.userKey.keyGen.signing.private.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.verify=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.verifyRecover=false
op.enroll.userKey.keyGen.signing.private.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.signing.privateKeyAttrId=k2
op.enroll.userKey.keyGen.signing.privateKeyNumber=2
op.enroll.userKey.keyGen.signing.public.keyCapabilities.decrypt=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.derive=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.encrypt=false
```

```
op.enroll.userKey.keyGen.signing.public.keyCapabilities.private=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.sensitive=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.sign=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.signRecover=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.token=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.unwrap=false
op.enroll.userKey.keyGen.signing.public.keyCapabilities.verify=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.verifyRecover=true
op.enroll.userKey.keyGen.signing.public.keyCapabilities.wrap=false
op.enroll.userKey.keyGen.signing.publicKeyAttrId=k3
op.enroll.userKey.keyGen.signing.publicKeyNumber=3
```

ピンリセット

ピンリセットは、合理的に実行されるべきかどうかを判断するポリシーに依存するため、ピンリセットの設定については 「トークンポリシー」 で説明しています。

更新

更新は、すでに登録されているトークンに対して実行することが合法であるかどうかを判断するためのポリシーに依存しているため、更新の設定については 「トークンポリシー」 で説明しています。

復元

TPS ユーザーインターフェイスのユーザーが以前にアクティブだったトークンを紛失や破棄などの好ましくない状態にマークすると、復元が暗黙的に開始されます。これが発生すると、同じユーザーによる次の新しいトークンの登録は、次の設定に従って、ユーザーの古いトークンからこの新しいトークンに証明書を復元します。

この操作の最終結果は、ユーザーが古いトークンから回復された暗号化証明書を含む可能性のある新しい物理トークンを取得することです。これにより、ユーザーは必要に応じてデータの暗号化と復号を続行できます。以下のサンプル設定例に示すように、通常、新しい署名証明書もこのトークンに配置されます。

以下は、設定に示されているように、TPS ユーザーインターフェイスでトークンを手動で配置できるサポートされている状態のリストです。

- **tokendb._069=#** - **DAMAGED (1)**: リカバリー設定の **destroyed** に相当します。トークンが物理的に破損された場合に使用します。
- **tokendb._070=#** - **PERM_LOST (2)**: リカバリー設定の **keyCompromise** に相当します。トークンが永久に失われた場合に使用されます。
- **tokendb._071=#** - **SUSPENDED (3)**: リカバリー設定の **onHold** に相当します。トークンを一時的に配置した際に使用されますが、ユーザーはトークンを再度検索することが予想されます。
- **tokendb._072=#** - **TERMINATED (6)**: リカバリー設定で **terminated** するもの。トークンをサービス対象外にするために内部の理由から使用します。

リカバリー設定の例:

```
#When a token is marked destroyed, don't revoke the certs on the token unless all other tokens do
not have the certs included:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.holdRevocationUntilLastCredential=false
```

```
#specify if we even want to revoke certs a token is marked destroyed:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeCert=false
#if we want to revoke any certs here, specify the reason for revocation that will be sent to the CA:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeCert.reason=0
#speficy if we want to revoke expired certs when marking the token destroyed:
op.enroll.userKey.keyGen.encryption.recovery.destroyed.revokeExpiredCerts=false
```

追加の設定を使用して、新しいトークンに対して回復操作を実行するときに (元のトークンが破棄済みとしてマークされている場合)、サポートされている静的回復の種類を指定します。以下のスキームがサポートされます。

- Recover Last (**RecoverLast**): トークンに配置される最新の暗号化証明書を復旧します。
- Generate New Key and Recover Last (**GenerateNewKeyAndRecoverLast**): Recover Last と同じですが、新しい暗号化証明書を生成し、トークンにアップロードします。新しいトークンには 2 つの証明書が含まれます。
- Generate New Key (**GenerateNewKey**): 新しい暗号化証明書を生成し、トークンに配置します。古い証明書は復元しないでください。

以下に例を示します。

```
op.enroll.userKey.keyGen.encryption.recovery.destroyed.scheme=RecoverLast
```

次の設定例は、永久に失われたとマークされたトークンを回復する方法を決定します。

```
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.holdRevocationUntilLastCredential=f
alse
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeCert.reason=1
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.revokeExpiredCerts=false
op.enroll.userKey.keyGen.encryption.recovery.keyCompromise.scheme=GenerateNewKey

# Section when a token is marked terminated.

op.enroll.userKey.keyGen.encryption.recovery.terminated.holdRevocationUntilLastCredential=false

op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeCert=true
op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeCert.reason=1
op.enroll.userKey.keyGen.encryption.recovery.terminated.revokeExpiredCerts=false
op.enroll.userKey.keyGen.encryption.recovery.terminated.scheme=GenerateNewKey

# This section details the recovery profile with respect to which certs and of what kind get
recovered on the token.

op.enroll.userKey.keyGen.recovery.destroyed.keyType.num=2
op.enroll.userKey.keyGen.recovery.destroyed.keyType.value.0=signing
op.enroll.userKey.keyGen.recovery.destroyed.keyType.value.1=encryption
```

最後に、次の例では、古いトークンにあった署名証明書に対してシステムが何を行うかを決定します。ほとんどの場合、署名秘密鍵の複数のコピーが使用可能になる可能性を回避するために、**GenerateNewKey** 復元スキームを使用する必要があります (たとえば、新しいトークンで復元されたものと、永久に失われたが他の誰かによって発見された古いトークンで復元されたもの)。

```
op.enroll.userKey.keyGen.recovery.keyCompromise.keyType.value.0=signing
```

```
op.enroll.userKey.keyGen.recovery.keyCompromise.keyType.value.1=encryption
op.enroll.userKey.keyGen.recovery.onHold.keyType.num=2
op.enroll.userKey.keyGen.recovery.onHold.keyType.value.0=signing
op.enroll.userKey.keyGen.recovery.onHold.keyType.value.1=encryption

op.enroll.userKey.keyGen.signing.recovery.destroyed.holdRevocationUntilLastCredential=false
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeCert.reason=0
op.enroll.userKey.keyGen.signing.recovery.destroyed.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.destroyed.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.holdRevocationUntilLastCredential=fals

op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeCert.reason=1
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.keyCompromise.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.onHold.holdRevocationUntilLastCredential=false
op.enroll.userKey.keyGen.signing.recovery.onHold.revokeCert=true

op.enroll.userKey.keyGen.signing.recovery.onHold.revokeCert.reason=6
op.enroll.userKey.keyGen.signing.recovery.onHold.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.onHold.scheme=GenerateNewKey
op.enroll.userKey.keyGen.signing.recovery.terminated.holdRevocationUntilLastCredential=false
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeCert=true
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeCert.reason=1
op.enroll.userKey.keyGen.signing.recovery.terminated.revokeExpiredCerts=false
op.enroll.userKey.keyGen.signing.recovery.terminated.scheme=GenerateNewKey

# Configuration for the case when we mark a token “onHold” or temporarily lost

op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.revokeCert=true
op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.revokeCert.reason=0
op.enroll.userKeyTemporary.keyGen.encryption.recovery.onHold.scheme=RecoverLast
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.num=2
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.value.0=signing
op.enroll.userKeyTemporary.keyGen.recovery.onHold.keyType.value.1=encryption
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.revokeCert=true
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.revokeCert.reason=0
op.enroll.userKeyTemporary.keyGen.signing.recovery.onHold.scheme=GenerateNewKey
```

アプレットの更新

以下の例は、Coolkey アプレット更新操作の設定方法を示しています。この操作は、フォーマット、登録、および PIN のリセット操作中に実行できます。

```
op.format.userKey.update.applet.directory=/usr/share/pki/tps/applets
op.format.userKey.update.applet.emptyToken.enable=true
op.format.userKey.update.applet.encryption=true
op.format.userKey.update.applet.requiredVersion=1.4.54de790f
```

これらのオプションの一部は、Format セクションですでに紹介されています。これらは、アプレットのアップグレードを許可する必要があるかどうか、アプレットファイルの場所、およびトークンをアップグレードするアプレットのバージョンを決定するために必要な情報を提供します。**requiredVersion** のバージョンは、**directory** 内のファイル名にマッピングされます。

キーの更新

この操作は、フォーマット、登録、および PIN リセット操作中に実行でき、ユーザーは Global Platform キーセットのバージョンを製造元が提供するデフォルトからアップグレードできます。

TPS

次のオプションは、特定のトークンに代わって要求された次のフォーマット操作中に、キーセットを 1 から 2 にアップグレードするように TPS に指示します。これが行われたら、TKS はトークンに書き込まれる 3 つの新しいキーを取得する必要があります。その後、トークンは同じ TPS および TKS インストールで使用する必要があります。そうしないと、ロックされます。

```
op.format.userKey.update.symmetricKeys.enable=true
op.format.userKey.update.symmetricKeys.requiredVersion=2
```

代わりに、現在よりも低いバージョンを指定して、キーセットをダウングレードすることもできます。

TKS

上記のように、TKS は、トークンに書き込む新しい鍵を生成するように設定する必要があります。まず、新しいマスターキー識別子 **02** は、次の例に示すように、TKS **CS.cfg** の PKCS #11 オブジェクトのニックネームにマップする必要があります。

```
tks.mk_mappings.#02#01=internal:new_master
tks.defKeySet.mk_mappings.#02#01=internal:new_master
```

上記の例では、キーセット番号が TKS NSS データベースに存在する実際のマスターキーにマップされます。

マスターキーは、**01** などの ID で識別されます。TKS は、この ID を、マッピングの **masterKeyId** 部分として指定した PKCS #11 オブジェクトのニックネームにマッピングします。したがって、最初の番号はマスターキーのバージョンが更新されると更新され、2 番目の番号は一貫性を保ちます。

バージョン 1 からバージョン 2 にアップグレードしようとすると、マッピングによって、新しいキーセットの 3 つの部分を取得するために使用されるマスターキーのニックネームを見つける方法が決まります。

上記の例 **internal** の設定は、マスターキーがあるトークンの名前を参照します。また、**nethsm** など、名前を持つ外部 HSM モジュールも使用できます。強力な **new_master** は、マスターキーのニックネーム自体の例です。

6.5. 内部登録

注記

一般情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『TPS Profiles』 セクションを参照してください。

内部登録. この場合、TPS プロファイル (トークンタイプ) は、 *マッピングリゾルバー* で決定されます。*外部の登録* とは対照的に、認証情報はプロファイル自体で定義されます。以下に例を示します。

```
op.enroll.userKey.auth.enable=true
op.enroll.userKey.auth.id=ldap1
```

外部登録とは、CA および KRA コネクター情報が各プロファイルの各キータイプで定義される点です。以下に例を示します。

```
op.enroll.userKey.keyGen.encryption.ca.conn=ca1
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
```

ただし、TKS コネクター情報はプロファイルごとに定義されます。

```
op.enroll.userKey.tks.conn=tks1
```

注記

内部登録および外部登録との間での登録タイプの切り替えにより、引き続き使用するには、以前に登録したトークンをすべてフォーマットする必要があります。

6.6. 外部登録

外部登録は、認証されたユーザーの LDAP レコードからトークンタイプ (TPS プロファイル) を取得します。また、証明書/鍵のリカバリー情報を同じユーザーレコードに指定できます。

External Registration TPS プロファイルは、前述の Internal Registration プロファイルと似ています。これにより、クライアント側の鍵とサーバー側の鍵の両方に新しい証明書登録を指定できます。内部登録とは異なり、トークンに取得および読み込む特定の証明書 (およびその一致する鍵) を選択できます。

注記

内部登録および外部登録との間での登録タイプの切り替えにより、引き続き使用するには、以前に登録したトークンをすべてフォーマットする必要があります。

6.6.1. 外部登録の有効化

外部登録は、TPS インスタンス全体に対してグローバルに有効にできます。以下の例は、外部登録に関連するグローバル設定パラメーターのセットを示しています。

```
externalReg.allowRecoverInvalidCert.enable=true
externalReg.authId=ldap1
externalReg.default.tokenType=externalRegAddToToken
externalReg.delegation.enable=true
externalReg.enable=true
externalReg.recover.byKeyID=false
externalReg.format.loginRequest.enable=true
externalReg.mappingResolver=keySetMappingResolver
```

6.6.2. ユーザー LDAP レコード属性名のカスタマイズ

次の例に、外部登録に関連する認証パラメーターを示します (デフォルト値を使用)。

```
auths.instance.ldap1.externalReg.certs.recoverAttributeName=certsToAdd
auths.instance.ldap1.externalReg.cuidAttributeName=tokenCUID
auths.instance.ldap1.externalReg.tokenTypeAttributeName=tokenType
```

LDAP レコード属性名はここでカスタマイズできます。ユーザーの LDAP レコードの実際の属性がこの設定と一致していることを確認します。

6.6.3. certsToAdd 属性の設定

certsToAdd 属性は、以下の形式で複数の値を取ります。

```
<cert serial # in decimal>,<CA connector ID>,<key ID>,<kra connector ID>
```

以下に例を示します。

```
59,ca1,0,kra1
```

重要

デフォルトでは、キーリカバリーは証明書ごとにキーを検索します。これにより、*<key ID>* 値が無関係になります。ただし、TPS はオプションでこの属性を使用してキーを検索するように設定できます。そのため、通常は値を 0 に設定するのは簡単です。この値は無効であるため、一致しないキーを取得できなくなります。

鍵 ID による復元は推奨されていません。これは、証明書がこの場合に鍵と一致しているかどうかを検証することができないためです。

証明書および CA 情報のみを持つ **certsToAdd** 属性を指定する場合、TPS は問題の証明書がトークン上にあり、保存する必要があることを仮定します。この概念は *キー保持* と呼ばれます。

以下の例は、ユーザー LDAP レコードに関連する属性を示しています。

```
tokenType: externalRegAddToToken
certstoadd: 59,ca1,0,kra1
certstoadd: 134,ca1,0,kra1
Certstoadd: 24,ca1
```

6.6.4. ユーザーマッチングの実施に対するトークン

必要に応じて、登録に使用されるトークンがユーザーレコードのトークンレコード固有の ID (CUID) 属性と一致するようにシステムを設定できます。この属性 (**tokencuid**) がレコードにない場合は、CUID 一致は強制されません。

```
Tokencuid: a10192030405028001c0
```

外部登録に関するもう 1 つの属性は、各トークンのトークンポリシーが無視されます。

注記

外部登録で回復される証明書とキーの場合は、CA と KRA のコネクター情報がユーザー LDAP レコードで指定されます。復元される証明書/キーに関連する TPS プロファイルで指定された CA または KRA コネクター情報、もしくはその両方は無視されます。

```
certstoadd: 59,ca1,0,kra1
```

6.6.5. 委譲サポート

委譲サポートは、認証 (ログイン)、データの暗号化と復号、または署名 (制限付き) に関してユーザーが代理で行動できる委任者がいる場合 (たとえば、会社の幹部に 1 人以上の委譲者がいる場合) に役立ちます。

シナリオの例としては、各幹部が、幹部に代わって行動するために使用する独自のトークンを持っている場合があります。このトークンには、(TPS プロファイルにより決定する) 以下の証明書と鍵の組み合わせが含まれます。

- Authentication certificate/keys: CN には、委譲の名前と一意の ID が含まれます。Subject Alternative Name (SAN) 拡張機能には、幹部の Principal Name (UPN) が含まれます。
- 暗号化証明書: ワイヤレスの暗号化証明書の正確なコピーです。
- 証明書の署名: CN には委譲の名前と一意の ID が含まれます。SAN には、エグゼクティブの RFC822Name が含まれています。

委譲サポートを有効にするには、以下のパラメーターを使用します。

```
externalReg.delegation.enable=true
```

重要

バグを回避するには、**/var/lib/pki/*instance_name*/tps/conf/CS.cfg** ファイルの ***op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId*** パラメーターを **caTokenUserDelegateAuthKeyEnrollment** に手動で設定します。

```
op.enroll.delegateISEtoken.keyGen.encryption.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

6.6.6. SAN および DN のパターン

認証インスタンス設定の **auths.instance.*<authID>*.ldapStringAttributes** は、認証中に取得する属性を指定します。以下に例を示します。

```
auths.instance.ldap1.ldapStringAttributes=mail,cn,uid,edipi,pcc,firstname,lastname,exec-edipi,exec-pcc,exec-mail,certsToAdd,tokenCUID,tokenType
```

ユーザーの LDAP レコードの取得後、これらの属性の値を参照して、**$auth.*<attribute name>*$** の形式で証明書の Subject Alternative Name (SAN) または Distinguished Name (DN) を形成することができます。以下に例を示します。

```
op.enroll.delegateIEtoken.keyGen.authentication.SANpattern=$auth.exec-edipi$.$auth.exec-pcc$@EXAMPLE.com
op.enroll.delegateIEtoken.keyGen.authentication.dnpattern=cn=$auth.firstname$.$auth.lastname$.$auth.edipi$,e=$auth.mail$,o=TMS Org
```

パターンが SAN および DN の TPS プロファイルで使用される場合は、TPS プロファイルに指定された CA 登録プロファイルが正しく設定されていることが重要です。以下に例を示します。

TPS、プロファイル delegateIEtoken で

```
op.enroll.delegateIEtoken.keyGen.authentication.ca.profileId=caTokenUserDelegateAuthKeyEnrollment
```

CA で、プロファイル caTokenUserDelegateAuthKeyEnrollment に登録します。

上記の TPS プロファイルで DN を指定できるようにするには、**subjectDNInputImpl** プラグインを入力のいずれかとして指定する必要があります。

```
input.i2.class_id=subjectDNInputImpl
input.i2.name=subjectDNInputImpl
```

同様に、上記の TPS プロファイルで SAN を指定できるようにするには、**subjectAltNameExtInputImpl** プラグインを指定する必要があります。

```
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl
```

subjAltExtpattern も指定する必要があります。

```
policyset.set1.p6.default.params.subjAltExtPattern_0=
(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.req_san_pattern_0$
```

上記の例では、OID **1.3.6.1.4.1.311.20.2.3** は User Principal Name (UPN) の OID で、**request.req_san_pattern_0** は、**delegateIEtoken** SAN パターンで指定された最初の SAN パターンになります。

複数の SAN を同時に指定できます。TPS 側で、複数の SAN をコンマ (",") で区切った **SANpattern** で指定します。CA 側では、対応する **subjAltExtPattern** の数を以下の形式で定義する必要があります。

```
policyset.<policy set id>.<policy id>.default.params.subjAltExtPattern_<ordered number>=
```

上記の例では、*<ordered number>* は 0 で始まり、TPS 側で指定した各 SAN パターンについて 1 つずつ増えます。

```
policyset.set1.p6.default.params.subjAltExtPattern_0=
policyset.set1.p6.default.params.subjAltExtPattern_1=
...
```

完全な例を以下に示します。

例6.1 SANpattern および DNpattern の設定

LDAP レコードには、以下の情報が含まれます。

```
givenName: user1a
mail: user1a@example.org
firstname: user1a
edipi: 123456789
pcc: AA
exec-edipi: 999999999
exec-pcc: BB
```

```
exec-mail: user1b@EXAMPLE.com
tokenType: delegateISEtoken
certstoadd: 59,ca1,0,kra1
```

TPS 外部登録プロファイル **delegateIEtoken** には、以下が含まれます。

- **SANpattern**:

```
op.enroll.delegateISEtoken.keyGen.authentication.SANpattern=$auth.exec-
edipi$.$auth.exec-pcc$@EXAMPLE.com
```

- **DNPattern**:

```
op.enroll.delegateISEtoken.keyGen.authentication.dnpattern=cn=$auth.firstname$.$auth.las
tname$.$auth.edipi$,e=$auth.mail$,o=TMS Org
```

CA **caTokenUserDelegateAuthKeyEnrollment** には以下が含まれます。

```
input.i2.class_id=subjectDNInputImpl
input.i2.name=subjectDNInputImpl
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl

policyset.set1.p6.constraint.class_id=noConstraintImpl
policyset.set1.p6.constraint.name=No Constraint
policyset.set1.p6.default.class_id=subjectAltNameExtDefaultImpl
policyset.set1.p6.default.name=Subject Alternative Name Extension Default
policyset.set1.p6.default.params.subjAltExtGNEnable_0=true
policyset.set1.p6.default.params.subjAltExtPattern_0=
(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.req_san_pattern_0$
policyset.set1.p6.default.params.subjAltExtType_0=OtherName
policyset.set1.p6.default.params.subjAltNameExtCritical=false
policyset.set1.p6.default.params.subjAltNameNumGNs=1
```

結果の証明書には次のものが含まれます。

```
Subject: CN=user1a..123456789,E=user1a@example.org,O=TMS Org
Identifier: Subject Alternative Name - 2.5.29.17
Critical: no
Value:
  OtherName: (UTF8String)1.3.6.1.4.1.311.20.2.3,999999999.BB@EXAMPLE.com
```

6.7. MAPPING RESOLVER の設定

Token Processing System は、デフォルトで単一のマッピングリゾルバーを提供します。リゾルバーは **FilterMappingResolver** と呼ばれます。本セクションでは、その設定について説明します。

注記

マッピングリゾルバーの一般的な情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『マッピングリゾルバー』セクションを参照してください。

6.7.1. Key Set Mapping Resolver

外部の登録時、ユーザーの認証を行う前に、キーセットはリゾルバーを使用して解決する必要があります。

キーセットマッピングリゾルバー名は以下のように定義されます。

```
externalReg.mappingResolver=<keySet mapping resolver name>
```

以下に例を示します。

```
externalReg.mappingResolver=keySetMappingResolver
```

以下の設定例は、完全なインスタンスの設定を示しています。

```
mappingResolver.keySetMappingResolver.class_id=filterMappingResolverImpl
mappingResolver.keySetMappingResolver.mapping.0.filter.appletMajorVersion=0
mappingResolver.keySetMappingResolver.mapping.0.filter.appletMinorVersion=0
mappingResolver.keySetMappingResolver.mapping.0.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenATR=
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenCUID.end=a1000000000000000000
mappingResolver.keySetMappingResolver.mapping.0.filter.tokenCUID.start=a0000000000000000000

mappingResolver.keySetMappingResolver.mapping.0.target.keySet=defKeySet
mappingResolver.keySetMappingResolver.mapping.1.filter.appletMajorVersion=1
mappingResolver.keySetMappingResolver.mapping.1.filter.appletMinorVersion=1
mappingResolver.keySetMappingResolver.mapping.1.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenATR=1234
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenCUID.end=
mappingResolver.keySetMappingResolver.mapping.1.filter.tokenCUID.start=
mappingResolver.keySetMappingResolver.mapping.1.target.keySet=defKeySet
mappingResolver.keySetMappingResolver.mapping.2.filter.appletMajorVersion=
mappingResolver.keySetMappingResolver.mapping.2.filter.appletMinorVersion=
mappingResolver.keySetMappingResolver.mapping.2.filter.keySet=
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenATR=
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenCUID.end=
mappingResolver.keySetMappingResolver.mapping.2.filter.tokenCUID.start=
mappingResolver.keySetMappingResolver.mapping.2.target.keySet=jForte
mappingResolver.keySetMappingResolver.mapping.order=0,1,2
```

上記の例は、**0**、**1**、および **2** という名前の 3 つのマッピングを定義します。これらは、例の **mappingResolver.keySetMappingResolver.mapping.order=0,1,2** 行を使用して昇順で順序付けされます。この順序は、入力パラメーターが最初にマッピングフィルター **0** に対して実行されることを意味します。入力パラメーターがそのフィルターに一致しない場合にのみ、マッピング順序の次のフィルターが試行されます。たとえば、以下の特性を持つトークンが評価されます。

```
CUID=a00000000000000000011
appletMajorVersion=0
appletMinorVersion=0
```

次に、マッピング **0** を渡し、そのターゲットが割り当てられます。これは、**defKeySet** に設定されます。これは、アプレットのバージョンが一致し、CUID がそのマッピングの CUID の開始範囲と終了範囲内にあるためです。

一方、トークンに以下のパラメーターがある場合には、以下を行います。

```
CUID=b00000000000000000000
ATR=2222
appletMajorVersion=1
appletMinorVersion=1
```

この場合、このトークンは指定された CUID 範囲外であるため、**0** のマッピングに失敗します。また、アプレットバージョンが一致すると ATR がマッピングされないため、**1** マッピングも失敗します。上記のトークンはマッピング **2** とそのターゲット **jForte** に割り当てられます。

マッピング **2** には、そのフィルターへの割り当てがないことに留意してください。これにより、マッピングがすべてのトークンと照合され、実質的にデフォルトの値になります。このようなマッピングは、マッピング順序の最後に指定する必要があります。これ以降、他のマッピングは評価されないためです。

6.7.2. Token Type (TPS) Mapping Resolver

Token Processing System で定義されたデフォルトの **tokenType** マッピングリゾルバーは、**formatProfileMappingResolver**、**enrollProfileMappingResolver**、および **pinResetProfileMappingResolver** の 3 つです。前のセクションで説明した外部登録の場合と比較すると、内部登録の場合、トークンタイプは実際には定義されたマッピングリゾルバーから計算されます。

トークンタイプマッピングリゾルバー名は以下のように定義されます。

```
op.<op>.mappingResolver=<mapping resolver name>
```

以下に例を示します。

```
op.enroll.mappingResolver=enrollProfileMappingResolver
```

以下の設定例は、**enrollProfileMappingResolver** を説明します。

```
mappingResolver.enrollProfileMappingResolver.class_id=filterMappingResolverImpl
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.appletMajorVersion=1
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenCUID.end=b1000000000000000000
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenCUID.start=b0000000000000000000
mappingResolver.enrollProfileMappingResolver.mapping.0.filter.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.0.target.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.appletMajorVersion=1
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenCUID.end=a0000000000000001000
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenCUID.start=a0000000000000000000
mappingResolver.enrollProfileMappingResolver.mapping.1.filter.tokenType=soKey
mappingResolver.enrollProfileMappingResolver.mapping.1.target.tokenType=soKey
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.appletMajorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.appletMinorVersion=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenATR=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenCUID.end=
```

```
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenCUID.start=
mappingResolver.enrollProfileMappingResolver.mapping.2.filter.tokenType=
mappingResolver.enrollProfileMappingResolver.mapping.2.target.tokenType=userKey
mappingResolver.enrollProfileMappingResolver.mapping.order=1,0,2
```

上記の例で、3 つのマッピングが **enrollProfileMappingResolver** で定義されます。マッピングの名前は **0**、**1**、および **2** です。**mappingResolver.enrollProfileMappingResolver.mapping.order=1,0,2** 行は、マッピングを処理する順序を定義します。トークンがマッピングと一致する場合、その順序でそれ以上のマッピングは評価されません。マッピングと一致しない場合は、順序の次のマッピングが試行されます。

以下のパラメーターが含まれるトークンの場合:

```
CUID=a0000000000000000011
appletMajorVersion=1
appletMinorVersion=0
extension: tokenType=soKey
```

アプレットバージョンが一致すると、CUID は指定された開始範囲および終了範囲内で失敗し、拡張機能 **tokenType** は一致するため、この設定を持つトークンはマッピング **1** 用のフィルターに一致します。そのため、このトークンには、そのマッピングのターゲットが割り当てられます (**soKey**)。

別の場合では、トークンに以下のパラメーターがある場合:

```
CUID=b0000000000000000010
appletMajorVersion=1
appletMinorVersion=1
```

この場合、CUID は指定された範囲外であるため、**1** トークンのマッピングに失敗します。**0** エクステンションがないため、**tokenType** へのマッピングも失敗します。以前のフィルターのいずれにも一致しないすべてのトークンに一致するフィルターが指定されていないため、このトークンはマッピング **2** に一致します。

6.8. 認証設定

Token Processing System は、デフォルトでユーザー ID とパスワード (**UidPwdDirAuthentication**) を使用したディレクトリーベースの認証をサポートします。認証インスタンスは、以下のパターンを使用して **CS.cfg** ファイルで定義されます。

```
auths.instance.<auths ID>.*
```

<auths ID> は、認証設定のために TPS プロファイルによって参照されるオーセンティケーターの名前です。以下に例を示します。

```
op.enroll.userKey.auth.id=ldap1
```

以下の設定例は、認証インスタンスの完全な定義を示しています。

```
auths.impl.UidPwdDirAuth.class=com.netscape.cms.authentication.UidPwdDirAuthentication
auths.instance.ldap1.pluginName=UidPwdDirAuth
auths.instance.ldap1.authCredName=uid
auths.instance.ldap1.dnpattern=
auths.instance.ldap1.externalReg.certs.recoverAttributeName=certsToAdd
auths.instance.ldap1.externalReg.cuidAttributeName=tokenCUID
```

```
auths.instance.ldap1.externalReg.tokenTypeAttributeName=tokenType
auths.instance.ldap1.ldap.basedn=dc=sjc,dc=example,dc=com
auths.instance.ldap1.ldap.ldapauth.authtype=BasicAuth
auths.instance.ldap1.ldap.ldapauth.bindDN=
auths.instance.ldap1.ldap.ldapauth.bindPWPrompt=ldap1
auths.instance.ldap1.ldap.ldapauth.clientCertNickname=subsystemCert cert-pki-tomcat
auths.instance.ldap1.ldap.ldapconn.host=host1.EXAMPLE.com
auths.instance.ldap1.ldap.ldapconn.port=389
auths.instance.ldap1.ldap.ldapconn.secureConn=False
auths.instance.ldap1.ldap.ldapconn.version=3
auths.instance.ldap1.ldap.maxConns=15
auths.instance.ldap1.ldap.minConns=3
auths.instance.ldap1.ldapByteAttributes=
auths.instance.ldap1.ldapStringAttributes=mail,cn,uid,edipi,pcc,firstname,lastname,exec-edipi,exec-
pcc,exec-mail,certsToAdd,tokenCUID,tokenType
auths.instance.ldap1.ldapStringAttributes._000=##################################
auths.instance.ldap1.ldapStringAttributes._001=# For isExternalReg
auths.instance.ldap1.ldapStringAttributes._002=#   attributes will be available as
auths.instance.ldap1.ldapStringAttributes._003=#       $<attribute>$
auths.instance.ldap1.ldapStringAttributes._004=#   attributes example:
auths.instance.ldap1.ldapStringAttributes._005=#mail,cn,uid,edipi,pcc,firstname,lastname,exec-
edipi,exec-pcc,exec-mail,certsToAdd,tokenCUID,tokenType
auths.instance.ldap1.ldapStringAttributes._006=##################################
auths.instance.ldap1.pluginName=UidPwdDirAuth
auths.instance.ldap1.ui.description.en=This authenticates user against the LDAP directory.
auths.instance.ldap1.ui.id.PASSWORD.credMap.authCred=pwd
auths.instance.ldap1.ui.id.PASSWORD.credMap.msgCred.extlogin=PASSWORD
auths.instance.ldap1.ui.id.PASSWORD.credMap.msgCred.login=password
auths.instance.ldap1.ui.id.PASSWORD.description.en=LDAP Password
auths.instance.ldap1.ui.id.PASSWORD.name.en=LDAP Password
auths.instance.ldap1.ui.id.UID.credMap.authCred=uid
auths.instance.ldap1.ui.id.UID.credMap.msgCred.extlogin=UID
auths.instance.ldap1.ui.id.UID.credMap.msgCred.login=screen_name
auths.instance.ldap1.ui.id.UID.description.en=LDAP User ID
auths.instance.ldap1.ui.id.UID.name.en=LDAP User ID
auths.instance.ldap1.ui.retries=3
auths.instance.ldap1.ui.title.en=LDAP Authentication
```

TPS 認証インスタンスは、CA の **UidPwdDirAuthentication** 認証インスタンスと同様に設定されます。これは、いずれも同じプラグインで処理されるためです。ただし、TPS では、CA 設定に加えて追加のパラメーターが必要になります。

(内部登録と外部登録の両方の) 共通操作の場合、この認証方法を呼び出すプロファイルは、クライアント側で UID とパスワードにラベルを付ける方法のプロジェクトを TPS が許可します。これは、上記の例の **auths.instance.ldap1.ui.id.UID.name.en=LDAP User ID** パラメーターおよび **auths.instance.ldap1.ui.id.PASSWORD.name.en=LDAP Password** パラメーターによって制御されます。この設定では、クライアントが UID/password ペアを LDAP User ID および LDAP Password として表示するように指示します。どちらのパラメーターもカスタマイズできます。

credMap.authCred エントリーは、内部認証プラグインが提示された情報を受け入れる方法を設定し、**credMap.msgCred** エントリーは、この情報が TPS に渡される方法を設定します。これらのフィールドでは、カスタマイズされたプラグインの実装を使用でき、カスタム認証プラグインを使用しない限り、デフォルト値のままにする必要があります。

外部登録に関連するパラメーターは、「外部登録」 を参照してください。

CA 認証設定と同様に、同じ認証実装に対して複数の認証インスタンスを定義できます。これは、TPS がユーザーの複数のグループを提供する場合に便利です。各グループに独自の TPS プロファイルを使用するよう指示することができます。各グループが独自のディレクトリーサーバー認証を使用するように設定されます。

6.9. コネクター

コネクターは、TPS が他のサブシステム (主に CA、KRA、および TKS) と通信する方法を定義します。通常、これらのパラメーターは TPS のインストール時に設定されます。以下は、コネクター設定の例になります。

```
tps.connector.ca1.enable=true
tps.connector.ca1.host=host1.EXAMPLE.com
tps.connector.ca1.maxHttpConns=15
tps.connector.ca1.minHttpConns=1
tps.connector.ca1.nickName=subsystemCert cert-pki-tomcat
tps.connector.ca1.port=8443
tps.connector.ca1.timeout=30
tps.connector.ca1.uri.enrollment=/ca/ee/ca/profileSubmitSSLClient
tps.connector.ca1.uri.getcert=/ca/ee/ca/displayBySerial
tps.connector.ca1.uri.renewal=/ca/ee/ca/profileSubmitSSLClient
tps.connector.ca1.uri.revoke=/ca/ee/subsystem/ca/doRevoke
tps.connector.ca1.uri.unrevoke=/ca/ee/subsystem/ca/doUnrevoke
tps.connector.kra1.enable=true
tps.connector.kra1.host=host1.EXAMPLE.com
tps.connector.kra1.maxHttpConns=15
tps.connector.kra1.minHttpConns=1
tps.connector.kra1.nickName=subsystemCert cert-pki-tomcat
tps.connector.kra1.port=8443
tps.connector.kra1.timeout=30
tps.connector.kra1.uri.GenerateKeyPair=/kra/agent/kra/GenerateKeyPair
tps.connector.kra1.uri.TokenKeyRecovery=/kra/agent/kra/TokenKeyRecovery
tps.connector.tks1.enable=true
tps.connector.tks1.generateHostChallenge=true
tps.connector.tks1.host=host1.EXAMPLE.com
tps.connector.tks1.keySet=defKeySet
tps.connector.tks1.maxHttpConns=15
tps.connector.tks1.minHttpConns=1
tps.connector.tks1.nickName=subsystemCert cert-pki-tomcat
tps.connector.tks1.port=8443
tps.connector.tks1.serverKeygen=true
tps.connector.tks1.timeout=30
tps.connector.tks1.tksSharedSymKeyName=sharedSecret
tps.connector.tks1.uri.computeRandomData=/tks/agent/tks/computeRandomData
tps.connector.tks1.uri.computeSessionKey=/tks/agent/tks/computeSessionKey
tps.connector.tks1.uri.createKeySetData=/tks/agent/tks/createKeySetData
tps.connector.tks1.uri.encryptData=/tks/agent/tks/encryptData
```

TPS プロファイルは、これらのコネクターを ID で参照します。たとえば、以下のようになります。

```
op.enroll.userKey.keyGen.signing.ca.conn=ca1
```

同じ種類のコネクター (複数の CA コネクターなど) を複数定義できます。これは、1つの TPS インスタンスが、異なるトークングループに複数のバックエンド Certificate System サーバーを提供する場合に便利です。

注記

TPS のコネクターの自動フェイルオーバーはサポートされていません。元のシステムのクローンが作成されていれば、TPS を別の CA、KRA、または TKS にポイントするには、手動フェイルオーバーの手順を実行する必要があります。

6.10. 失効ルーティングの設定

失効ルーティングを設定するには、まず関連する CA コネクターのリストを定義し、それらを以下の形式でコネクターリストに追加する必要があります。

```
tps.connCAList=ca1,ca2
```

さらに、CA 署名証明書を TPS **nssdb** に追加し、信頼を設定する必要があります。

```
#cd <TPS instance directory>/alias
```

```
#certutil -d . -A -n <CA signing cert nickname> -t “CT,C,C” -i <CA signing cert b64 file name>
```

最後に、以下のオプションを使用して CA 署名証明書のニックネームをコネクターに追加する必要があります。

```
tps.connector.ca1.caNickname=caSigningCert cert-pki-tomcat CA
```

注記

CA の検出時に、TPS は CA の Authority Key Identifier を自動的に計算し、コネクター設定に追加する場合があります。以下に例を示します。

```
tps.connector.ca1.caSKI=i9wOnN0QZLkzkndAB1MKMcjbRP8=
```

この動作は想定されています。

6.11. サーバー側の鍵生成の設定

サーバー側の鍵の生成は、鍵が任意の Certificate System サブシステムである Key Recovery Authority (KRA) により生成されることを意味します。KRA によるキーの生成は、紛失したトークンまたは破損したトークンのキーの回復、または外部登録の場合のキーの取得を可能にするために必要です。本セクションでは、TMS でサーバー側の鍵の生成を設定する方法を説明します。

TPS のインストール時に、キーのアーカイブを使用するかどうかを尋ねられます。確認すると、セットアップは自動基本設定、特に次のパラメーターを実行します。

KRA の TPS コネクターパラメーター:

```
tps.connector.kra1.enable=true
tps.connector.kra1.host=host1.EXAMPLE.com
tps.connector.kra1.maxHttpConns=15
tps.connector.kra1.minHttpConns=1
tps.connector.kra1.nickName=subsystemCert cert-pki-tomcat
tps.connector.kra1.port=8443
```

```
tps.connector.kra1.timeout=30
tps.connector.kra1.uri.GenerateKeyPair=/kra/agent/kra/GenerateKeyPair
tps.connector.kra1.uri.TokenKeyRecovery=/kra/agent/kra/TokenKeyRecovery
```

サーバー側の鍵生成用の TPS プロファイル固有のパラメーター:

```
op.enroll.userKey.keyGen.encryption.serverKeygen.archive=true
op.enroll.userKey.keyGen.encryption.serverKeygen.drm.conn=kra1
op.enroll.userKey.keyGen.encryption.serverKeygen.enable=true
```

serverKeygen.archive の **serverKeygen.enable=true** オプションを有効にします。

重要

LunaSA HSM は、RSA 暗号化用に 2048 ビットより小さい鍵サイズに対応しません。

たとえば、鍵のサイズを 2048 ビットに設定するには、**/var/lib/pki/*instance_name*/tps/conf/CS.cfg** ファイルに以下のパラメーターを設定します。

```
op.enroll.userKey.keyGen.encryption.keySize=2048
```

TKS 設定:

以下は、(TPS を介して) TKS と KRA との間の通信に使用されるトランスポート証明書のニックネームを設定します。

```
tks.drm_transport_cert_nickname=transportCert cert-pki-tomcat KRA
```

参照したトランスポート証明書も TKS インスタンスセキュリティーモジュールに存在する必要があります。以下に例を示します。

```
transportCert cert-pki-tomcat KRA                u,u,u
```

KRA の設定

PKCS#11 トークンに応じて、**kra.keygen.temporaryPairs** パラメーター、**kra.keygen.sensitivePairs** パラメーター、および **kra.keygen.extractablePairs** パラメーターは、鍵生成オプションに合わせてカスタマイズできます。これらのパラメーターはすべて、デフォルトで **false** に設定されます。

これらのパラメーターの値は、Red Hat Certificate System でサポートされているセキュリティーモジュールでテストされています。

NSS (FIPS モードの場合):

```
kra.keygen.extractablePairs=true
```

nCipher neld Connect 6000 (デフォルトでは指定なしの機能)

RSA 鍵を指定する場合:

```
kra.keygen.temporaryPairs=true
```

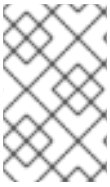
(他のパラメーターは指定しないでください。)

ECC キーを生成する場合:

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=false
kra.keygen.extractablePairs=true
```

LunaSA CKE - Key Export Model (非 FIPS モード):

```
kra.keygen.temporaryPairs=true
kra.keygen.sensitivePairs=true
kra.keygen.extractablePairs=true
```



注記

Gemalto SafeNet LunaSA は、CKE - Key Export モデルでの PKI 秘密鍵抽出のみおよび非 FIPS モードでのみサポートされます。FIPS モードの LunaSA Cloning モデルおよび CKE モデルは、PKI 秘密鍵の抽出をサポートしません。

注記

LunaSA CKE - Key Export Model が FIPS モードにあると、pki 秘密鍵を抽出できません。

6.12. 新しい鍵セットの設定

本セクションでは、Token Processing System (TPS) および Token Key Service (TKS) で設定したデフォルトのキーの代わりに設定する方法を説明します。

TKS 設定

デフォルトのキーセットは、**/var/lib/pki/*instance_name*/tks/conf/CS.cfg** ファイルで以下のオプションを使用して TKS に設定されます。

```
tks.defKeySet._000=##
tks.defKeySet._001=## Axalto default key set:
tks.defKeySet._002=##
tks.defKeySet._003=## tks.defKeySet.mk_mappings.#02#01=<tokenname>:<nickname>
tks.defKeySet._004=##
tks.defKeySet.auth_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.kek_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.mac_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.defKeySet.nistSP800-108KdfOnKeyVersion=00
tks.defKeySet.nistSP800-108KdfUseCuidAsKdd=false
```

上記の設定は、TMS で使用できる特定のタイプまたはクラスのトークンに固有の設定を定義します。最も重要な部分は、3 つの開発者キーまたは (すぐに使用できる) セッションキーです。これらは、対称キーのハンドオーバーが行われる前に安全なチャネルを作成するために使用されます。これらのキーのタイプが異なる場合には、これらのキーに異なるデフォルト値が設定される可能性があります。

nistSP800 キー分散方式を記述した設定では、この方式を使用するか、標準的な Visa 方式を使用するかを制御します。具体的には、**tks.defKeySet.nistSP800-108KdfOnKeyVersion** オプションの値により NIST バージョンが使用されることが判断されます。この **nistSP800-108KdfUseCuidAsKdd** オプションを使用すると、処理時に CUID のレガシーキー ID 値を使用できます。新しい KDD 値が最も一般的に使用されるため、このオプションはデフォルトで無効 (**false**) になります。これにより、新しいキーセットを設定して、新しいクラスのキーのサポートを有効にすることができます。

例6.2 jForte クラスのサポートの有効化

jForte クラスのサポートを有効にするには、以下を設定します。

```
tks.jForte._000=##
tks.jForte._001=## SAFLink's jForte default key set:
tks.jForte._002=##
tks.jForte._003=## tks.jForte.mk_mappings.#02#01=<tokenname>:<nickname>
tks.jForte._004=##
tks.jForte.auth_key=#30#31#32#33#34#35#36#37#38#39#3a#3b#3c#3d#3e#3f
tks.jForte.kek_key=#50#51#52#53#54#55#56#57#58#59#5a#5b#5c#5d#5e#5f
tks.jForte.mac_key=#40#41#42#43#44#45#46#47#48#49#4a#4b#4c#4d#4e#4f
tks.jForte.nistSP800-108KdfOnKeyVersion=00
tks.jForte.nistSP800-108KdfUseCuidAsKdd=false
```

以前の例と比較して、3 つの静的セッションキーの違いに注意してください。

Certificate System は、Giesecke & Devrient (G&D) Smart Cafe 6 スマートカードの Secure Channel Protocol 03 (SCP03) をサポートします。TKS でこれらのスマートカードに対する SCP03 サポートを有効にするには、**/var/lib/pki/*instance_name*/tks/conf/CS.cfg** ファイルに設定します。

```
tks.defKeySet.prot3.divers=emv
tks.defKeySet.prot3.diversVer1Keys=emv
tks.defKeySet.prot3.devKeyType=DES3
tks.defKeySet.prot3.masterKeyType=DES3
```

TPS 設定

サポートしているクライアントがトークンで操作を実行しようとすると、TPS が新しいキーセットを認識するように設定する必要があります。デフォルトの **defKeySet** は、ほとんどの場合使用されます。

TPS で **keySet** を決定する主な方法は、「Mapping Resolver の設定」 を決定します。このリゾルバーメカニズムを確立するために必要な正確な設定については、リンクされたセクションを参照してください。

KeySet Mapping Resolver が存在しない場合は、TPS が適切な **keySet** を決定するのに複数のフォールバック方法を使用できます。

- TPS の **CS.cfg** 設定ファイルに、**tps.connector.tks1.keySet=defKeySet** を追加できます。
- 特定のクライアントは、希望する **keySet** 値を明示的に渡すように設定することもできます。ただし、現時点では、Enterprise Security Client にはこの機能はありません。
- TPS が希望の方法に基づいて適切な **keySet** を計算すると、セキュアなチャネルの作成にも **keySet** 値を渡すことができるように TKS へのすべての要求を計算します。その後、TKS は (上記の説明) 独自の **keySet** 設定を使用し、続行方法を決定します。

6.13. 新しいマスターキーの設定

本セクションでは、Token Key Service (TKS) で新しいマスターキーを設定するのに必要な手順および設定を説明します。背景情報は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 を参照してください。

手順6.1 新規マスターキーの作成

1. TKS セキュリティーデータベースへのアクセスに必要な PIN の内部を取得します。

```
# cat /var/lib/pki/pki-tomcat/tks/conf/password.conf
internal=649713464822
internaldb=secret12
replicationdb=-752230707
```

2. TKS インスタンスの **alias/** ディレクトリーを開きます。

```
# cd /var/lib/pki/pki-tomcat/alias
```

3. **tkstool** ユーティリティーを使用して新規マスターキーを生成します。以下に例を示します。

```
# tkstool -M -n new_master -d /var/lib/pki/pki-tomcat/alias -h <token_name>
Enter Password or Pin for "NSS Certificate DB":

Generating and storing the master key on the specified token . . .

Naming the master key "new_master" . . .

Computing and displaying KCV of the master key on the specified token . . .

new_master key KCV:  CA5E 1764
```

4. 鍵がデータベースに正しく追加されていることを確認します。

```
# tkstool -L -d .


 slot:  NSS User Private Key and Certificate Services
token:  NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
       <0> new_master
```

6.13.1. ラップされたマスターキーの生成および転送 (Key Ceremony)

マスターキーを外部トークンまたは複数の場所で使用する場合は、ハードウェアトークンに安全に転送できるように **ラップ** する必要があります。この **tkstool** ユーティリティーを使用するとトランスポートキーを生成できます。次に、トークンが生成されるファシリティーにマスターキーを送信します。ラップマスターキーを転送するプロセスは、一般的に *キーセレモニー* と呼ばれます。

注記

トランスポートキーは、生成されたマスターキーとのみ使用できます。

手順6.2 ラップされたマスターキーの生成および転送

1. Token Key Service セキュリティーデータベースへのアクセスに必要な内部 PIN を取得します。

```
# cat /var/lib/pki/pki-tomcat/tks/conf/password.conf

internal=649713464822
internaldb=secret12
replicationdb=-752230707
```

2. TKS インスタンスの **alias**/ ディレクトリーを開きます。

```
# cd /var/lib/pki/pki-tomcat/alias
```

3. **transport** という名前のトランスポートキーを作成します。

```
# tkstool -T -d . -n transport
```

注記

tkstool ユーティリティーは、生成された 3 つのセッションキーごとにキー共有と KCV 値を出力します。この手順の後半で新しいデータベースにトランスポートキーを再生成する必要がある場合に、それらをファイルに保存し、失われた場合はキーを再生成します。

4. プロンプトが表示されたら、データベースのパスワードを入力します。次に、画面の指示に従って、ランダムなシードを生成します。

```
A random seed must be generated that will be used in the
creation of your key.  One of the easiest ways to create a
random seed is to use the timing of keystrokes on a keyboard.

To begin, type keys on the keyboard until this progress meter
is full.  DO NOT USE THE AUTOREPEAT FUNCTION ON YOUR KEYBOARD!


Continue typing until the progress meter is full:

|************************************************************|

Finished.


Type the word "proceed" and press enter
```

5. 次のプロンプトにより、一連のセッションキーが生成されます。最終メッセージになるまで、画面の指示に従ってください。

```
Successfully generated, stored, and named the transport key!
```

6. トランスポートキーを使用してマスターキーを生成してラップし、これを **file** という名前のファイルに保存します。

```
# tkstool -W -d . -n new_master -t transport -o file
Enter Password or Pin for "NSS Certificate DB":
Retrieving the transport key (for wrapping) from the specified token . . .
Generating and storing the master key on the specified token . . .
Naming the master key "new_master" . . .
Successfully generated, stored, and named the master key!
Using the transport key to wrap and store the master key . . .
Writing the wrapped data (and resident master key KCV) into the
 file called "file" . . .

      wrapped data:   47C0 06DB 7D3F D9ED
                      FE91 7E6F A7E5 91B9
      master key KCV: CED9 4A7B
      (computed KCV of the master key residing inside the wrapped data)
```

7. ラップされたマスターキーを適切な場所またはファシリティーにコピーします。

8. 必要な場合は、HSM またはファシリティーで新しいセキュリティーデータベースを生成します。

```
# tkstool -N -d <directory>
```

新たなデータベースで生成した鍵と同じ鍵を生成する **-I** オプションを追加します。この方法でトランスポートキーを再生成するには、この手順で前のステップで生成した各セッションキーに対してセッションキー共有と KCV を入力する必要があります。

```
# tkstool -I -d <directory> -n verify_transport
```

9. トランスポートキーを使用して、ファイルに保存されているマスターキーをアンラップします。要求されたら、セキュリティーデータベースの PIN を入力します。

```
# tkstool -U -d directory -n new_master -t verify_transport -i file
Enter Password or Pin for "NSS Certificate DB":
Retrieving the transport key from the specified token (for
 unwrapping) . . .
Reading in the wrapped data (and resident master key KCV) from
 the file called "file" . . .

    wrapped data:   47C0 06DB 7D3F D9ED
                    FE91 7E6F A7E5 91B9
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to temporarily unwrap the master key to
recompute its KCV value to check against its pre-computed KCV value . . .
    master key KCV: CED9 4A7B
    (computed KCV of the master key residing inside the wrapped data)
    master key KCV: CED9 4A7B
    (pre-computed KCV of the master key residing inside the wrapped data)

Using the transport key to unwrap and store the master key on the
 specified token . . .
Naming the master key "new_master" . . .
Successfully unwrapped, stored, and named the master key!
```

10. 鍵がデータベースに追加されたことを確認します。

```
# tkstool -L -d
slot:  NSS User Private Key and Certificate Services
token:  NSS Certificate DB

Enter Password or Pin for "NSS Certificate DB":
   <0> transport
   <1> new_master
```

6.14. TKS/TPS 共有対称キーの設定

共有対称キーは、TPS サブシステムと TKS サブシステムの両方の **NSS** データベースに存在する必要があります。このキーは、TPS サブシステムの作成時に自動的に生成されます。TPS と TKS の両方が同じ Tomcat インスタンスにインストールされている場合は、TKS により自動的に鍵を使用するため、追加の設定は必要ありません。ただし、両方のサブシステムが別のインスタンス上にある場合や、別の物理ホストがある場合は、本セクションで説明されている手順に従って鍵を TKS に安全に転送する必要があります。

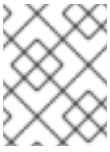
TPS と TKS との間で共有キーを安全に転送するには、いくつかの方法を使用できます。

- authomatic メソッド: このメソッドは、TPS のサブシステム証明書がソフトウェア NSS データベースに保持されている場合に機能します。
- 上記の方法が失敗した場合は、フォールバック手動の方法を使用できます。この方法では、TPS からキーをラップできる **tkstool** ユーティリティーを使用して TPS で共有キーを生成できます。これにより、転送中にキーを公開せずに安全な転送を可能にし、TKS NSS データベースにアンラップします。

以下は、鍵のインポートに使用される方法に関係なく、TPS と TKS の両方の一般的な設定を説明します。自動方式により、これらの設定が自動的に生成されることに注意してください。

TKS

```
tks.useNewSharedSecretNames=true
tps.0.host=dhcp-16-206.sjc.example.com
tps.0.nickname=TPS-<tps host name>-8443 sharedSecret
tps.0.port=8443
tps.0.userid=,TPS-<tps host name>-8443
tps.list=0
```

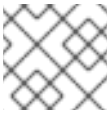


注記

上記のリストは、TKS が複数の TPS インスタンスに接続している場合に拡張できます。

TPS

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```



注記

ホスト名は、TKS 側で設定されるホスト名と同じでなければなりません。

6.14.1. 共有対称キーを手動で生成して転送

本セクションでは、共有対称鍵を手動で生成および転送する方法を説明します。この方法は、自動生成およびトランスポートが失敗した場合に便利ですが、それ以外の場合には回避する必要があります。

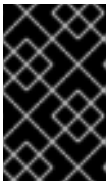
手動の方法は、2 つの手順で設定されています。最初のインスタンスは Token Key Service 側で実行され、トークン処理システムの 2 番目のサービスで実行されます。

手順6.3 手動共有シークレットキーメソッド - TKS 側

1. 最初のシステムにトークンキーサービスをインストールします。インストール手順については、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 を参照してください。
2. TKS サービスを停止します。

```
#pki-server stop pki-tomcat
```

3. **/var/lib/pki/pki-tomcat/alias** ディレクトリーに移動し、**tkstool** を使用して TKS で共有の秘密鍵を作成します。新しい TKS インスタンスを再起動する前に、共有鍵を生成してください。



重要

tkstool スクリプトにより、キーの作成プロセス中に鍵に関する情報が表示されます。後でキーを TPS にインポートするのに必要となるため、この情報を書き留めてください。

```
#cd /var/lib/pki/pki-tomcat/alias
#tkstool -T -d /var/lib/pki/pki-tomcat/tks/alias -n TPS-<tps host name>-8443 sharedSecret
Generating the first session key share . . .
   first session key share:      792F AB89 8989 D902
                                 9429 6137 8632 7CC4
   first session key share KCV:  D1B6 14FD
Generating the second session key share . . .
   second session key share:     4CDF C8E0 B385 68EC
                                 380B 6D5E 1C19 3E5D
   second session key share KCV:  1EC7 8D4B
Generating the third session key share . . .
   third session key share:      CD32 3140 25B3 C789
                                 B54F 2C94 26C4 9752
   third session key share KCV:  73D6 8633
Generating first symmetric key . . .
Generating second symmetric key . . .
Generating third symmetric key . . .
Extracting transport key from operational token . . .
   transport key KCV:  A8D0 97A2
Storing transport key on final specified token . . .
Naming transport key "sharedSecret" . . .
Successfully generated, stored, and named the transport key!
```

4. TKS に新しいキーを設定します。

```
tks.useNewSharedSecretNames=true
tps.0.host=dhcp-16-206.sjc.redhat.com
```

```
tps.0.nickname=TPS-<tps host name>-8443 sharedSecret
tps.0.port=8443
tps.0.userid=TPS-<tps host name>-8443 sharedSecret
tps.list=0
```

5. TKS を起動します。

```
#pki-server start pki-tomcat
```

手順6.4 手動による共有シークレットキーメソッド - TPS 側

1. 2 番目のシステムに Token Processing System をインストールします。インストール手順については、『Red Hat Certificate System 10 計画、インストール、およびデプロイメントのガイド』 を参照してください。
2. TPS サービスを停止します。

```
#pki-server stop pki-tomcat
```

3. **/var/lib/pki/pki-tomcat/alias** ディレクトリーに移動し、**tkstool** を使用して NSS ソフトウェアトークンに共有鍵をインポートします。

```
#cd /var/lib/pki/pki-tomcat/alias
#tkstool -I -d . -n TPS-<tps host name>-8443 sharedSecret
```

この時点で、スクリプトは上記の手順の TKS 側で共有鍵を生成およびラップする際に表示されるセッションキー共有を求めるプロンプトを表示します。

4. TPS で共有シークレットを設定します。

```
conn.tks1.tksSharedSymKeyName=TPS-<tps host name>-8443 sharedSecret
```

5. TPS サービスを起動します。

```
#pki-server start pki-tomcat
```

6.15. 異なる SCP バージョンでの異なるアプレットの使用

Certificate System では、**/var/lib/*instance_name*/tps/conf/CS.cfg** ファイルの以下のパラメーターで、各トークン操作に対してすべての Secure Channel Protocol (SCP) バージョンに読み込むアプレットを指定します。

```
op.operation.token_type.update.applet.requiredVersion=version
```

ただし、以下のパラメーターを追加して、特定の SCP バージョンに個別のアプレットを設定することもできます。

```
op.operation.token_type.update.applet.requiredVersion.prot.protocol_version=version
```

Certificate System は、以下の操作の個別のプロトコルバージョンの設定に対応します。

- **format**

- **enroll**
- **pinReset**

例6.3 登録操作用のプロトコルバージョンの設定

userKey トークンの登録操作を実行するときに、SCP03 に特定のアプレットを設定し、他のすべてのプロトコルに別のアプレットを設定するには、以下を行います。

1. **/var/lib/*instance_name*/tps/conf/CS.cfg** ファイルを編集します。
   a. デフォルトで使用されるアプレットを指定するには、***op.enroll.userKey.update.applet.requiredVersion*** パラメーターを設定します。以下に例を示します。

      ```
      op.enroll.userKey.update.applet.requiredVersion=1.4.58768072
      ```

   b. ***op.enroll.userKey.update.applet.requiredVersion.prot.3*** パラメーターを設定して、アプレットの Certificate System が SCP03 プロトコルに使用するように設定します。以下に例を示します。

      ```
      op.enroll.userKey.update.applet.requiredVersion.prot.3=1.5.558cdcff
      ```

2. Certificate System を再起動します。

   ```
   pki-server restart instance_name
   ```

TKS で SCP03 for Giesecke & Devrient (G&D) Smart Cafe 6 スマートカードを有効にする方法は、「新しい鍵セットの設定」 を参照してください。

第7章 証明書の取り消しおよび CRL 発行

Certificate System は、証明書の取り消しと、失効した証明書一覧 (CRL) と呼ばれる失効証明書の一覧を生成する方法を提供します。この章では、証明書を取り消す方法と、CMC の取り消しについて説明し、CRL と CRL 設定について詳しく説明します。

7.1. 証明書の失効について

証明書は、エンドユーザー (証明書の元の所有者) または Certificate Manager エージェントによって取り消すことができます。エンドユーザーは、エンドエンティティーページにある失効フォームを使用して証明書を取り消すことができます。エージェントは、エージェントサービスインターフェイスで適切な形式を使用して、エンドエンティティー証明書を破棄することができます。いずれの場合も、証明書ベース (SSL/TLS クライアント認証) が必要です。

エンドユーザーは、認証用に提示された証明書と同じサブジェクト名が含まれる証明書のみを取り消します。認証に成功すると、サーバーはエンドユーザーに属する証明書を一覧表示します。次に、エンドユーザーが失効する証明書を選択するか、リスト内のすべての証明書を取り消します。エンドユーザーは、各証明書またはリスト全体の失効日や失効理由などの追加の詳細を指定することもできます。

エージェントは、シリアル番号の範囲や発行先名コンポーネントに基づいて証明書を取り消します。失効要求が送信されると、エージェントは、失効する証明書を選択できる証明書のリストを受け取ります。エージェントがエンドユーザーの証明書を破棄する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』を参照してください。

失効要求が承認されると、Certificate Manager は、その内部データベースで対応する証明書レコードを失効し、これを設定すると、公開ディレクトリーから失効した証明書が削除されます。これらの変更は、CA が発行する次の CRL に反映されます。

ID トークンとして公開鍵証明書を使用するサーバーおよびクライアントアプリケーションには、証明書の有効性に関する情報へのアクセスが必要です。証明書の有効性を決定する要素の 1 つが失効ステータスであるため、これらのアプリケーションは、検証する証明書が取り消されているかどうかを確認する必要があります。CA は以下を行う責任があります。

- 失効要求が CA によって受け取られ、承認されている場合は、証明書を取り消します。
- 失効した証明書のステータスを、その有効性ステータスを確認する必要のある関係者またはアプリケーションが利用できるようにします。

証明書が取り消されるたびに、Certificate Manager は内部データベース内の証明書のステータスを自動的に更新し、内部データベース内の証明書のコピーを失効としてマークし、データベースから証明書を削除するように Certificate Manager が設定されている場合は、失効した証明書を公開ディレクトリーから削除します。

証明書の失効ステータスを通信するための標準的な方法の 1 つは、失効した証明書のリスト (証明書失効リスト (CRL)) を公開することです。CRL は、失効した証明書の公開されている証明書の公開されているリストです。

Certificate Manager は CRL を生成するように設定できます。これらの CRL は、CRL 設定で拡張固有のモジュールを有効にすることで、X.509 標準に準拠するように作成できます。サーバーは、CRL 発行ポイントフレームワークを介して標準の CRL 拡張をサポートします。発行ポイントの CRL 拡張設定の詳細は、「CRL 拡張機能の設定」 を参照してください。証明書マネージャーは、証明書が取り消されるたびに、定期的に CRL を生成できます。公開が設定されている場合、CRL はファイル、LDAP ディレクトリー、または OCSP レスポンダーに公開できます。

CRL は、CRL にリストされている証明書を発行した CA によって、またはその CA によって CRL の発行を許可されたエンティティーによって発行され、デジタル署名されます。CA は、単一のキーペアを

使用して、発行する証明書と CRL の両方に署名することも、2 つのキーペアを、1つは発行する証明書、もう1つは CRL にそれぞれ使用することもできます。

デフォルトでは、Certificate Manager は1つのキーペアを使用して、発行する証明書を署名し、生成する CRL を生成します。Certificate Manager に別のキーペアを作成し、CRL の署名専用に使用する場合は、「異なる証明書を使用するように CA を設定して CRL を署名」 を参照してください。

CRL は、発行ポイントが定義および設定されているとき、および CRL 生成が有効になっているときに生成されます。

CRL が有効になっている場合、サーバーは証明書が取り消されるときに失効情報を収集します。サーバーは、設定されたすべての発行ポイントに対して、取り消された証明書との一致を試みます。特定の証明書は、どの発行ポイントとも一致できないか、1つの発行ポイント、複数の発行ポイント、またはすべての発行ポイントと一致します。取り消された証明書が発行ポイントと一致すると、サーバーは証明書に関する情報をその発行ポイントのキャッシュに格納します。

キャッシュをコピーする間隔は秒単位で内部ディレクトリーにコピーされます。CRL を作成する間隔に達すると、CRL がキャッシュから作成されます。この発行ポイントにデルタ CRL が設定されている場合は、この時点でデルタ CRL も作成されます。Certificate Manager がこの情報の収集を開始したため、完全な CRL には、失効した証明書情報がすべて含まれます。デルタ CRL には、完全な CRL の最終更新以降、取り消されたすべての証明書情報が含まれます。

完全な CRL は、デルタ CRL のように順番に番号が付けられます。完全な CRL とデルタは同じ番号を持つことができます。この場合、デルタ CRL は **次** の完全な CRL と同じ番号を持ちます。たとえば、完全な CRL が最初の CRL の場合、これは CRL 1 になります。デルタ CRL は Delta CRL 2 です。CRL 1 と Delta CRL 2 で結合されたデータは、次の完全な CRL である CRL 2 と同等です。

注記

発行ポイントの拡張に変更を加えると、その発行ポイントに対して次の完全な CRL でデルタ CRL が作成されません。デルタ CRL は、作成される **2 番目** の完全な CRL で作成され、その後のすべての完全な CRL と共に作成されます。

内部データベースには、最新の CRL および delta CRL のみが保存されます。新しい CRL が作成されると、古い CRL が上書きされます。

CRL が公開されると、CRL およびデルタ CRL の各更新は、公開設定で指定された場所に公開されます。公開する方法は、保存される CRL の数を決定します。ファイル公開では、各 CRL は、CRL の番号を使用してファイルに公開されるため、ファイルは上書きされません。LDAP 公開では、公開される各 CRL はディレクトリーエントリーに CRL を含む属性の古い CRL に置き換えられます。

デフォルトでは、CRL には失効した証明書に関する情報が含まれません。サーバーには、発行ポイントにオプションを有効にして、失効した証明書を含めることができます。期限切れの証明書が含まれている場合、失効した証明書に関する情報は、証明書の有効期限が切れても CRL から削除されません。失効した証明書が含まれていない場合は、証明書の有効期限が切れると、失効した証明書に関する情報が CRL から削除されます。

7.1.1. ユーザーが開始した失効

エンドユーザーが証明書失効要求を送信すると、失効プロセスの最初のステップは、Certificate Manager がエンドユーザーを識別および認証して、ユーザーが他の誰かに属する証明書ではなく、自分の証明書を失効させようとしていることを確認することです。

SSL/TSL クライアント認証では、サーバーは、エンドユーザーが取り消されるものと同じサブジェクト名を持つ証明書を提示することを期待し、それを認証目的で使用します。サーバーは、クライアント認証用に提示された証明書のサブジェクト名を内部データベースの証明書にマッピングすることによ

り、失効要求の信頼性を検証します。サーバーは、証明書が内部データベース内で1つ以上の有効な証明書にマップされている場合に限り証明書を取り消します。

認証に成功すると、サーバーは、クライアント認証に対して提示される証明書の発行先名と一致する、有効または期限切れの証明書の一覧を表示します。次に、ユーザーは、取り消す証明書を選択するか、リスト内のすべての証明書を取り消すことができます。

7.1.2. 証明書の失効理由

Certificate Manager は、発行した証明書をすべて取り消すことができます。次のように、CRL に含まれることが多い証明書を取り消すための一般的に受け入れられている理由コードがあります。

- **0**.不特定 (特に理由はありません)。
- **1**.証明書に関連する秘密鍵が侵害されました。
- **2**.証明書を発行した CA に関連付けられた秘密鍵が危険にさらされました。
- **3**.証明書の所有者は、証明書の発行者とは関係がなくなり、その証明書で取得したアクセス権を失ったか、証明書を必要としなくなりました。
- **4**.別の証明書がこれに置き換わります。
- **5**.証明書を発行した CA は、操作する予定です。
- **6**.証明書は、さらなるアクションが行われるまで保留されています。これは取り消されたものとして処理されますが、証明書がアクティブで再度有効になるように、将来保持されないことがあります。
- **8**.証明書は保留から削除されたため、CRL から **削除** されます。これはデルタ CRL でのみ有効です。
- **9**.証明書の所有者の権限が撤回されているため、証明書は取り消されます。

証明書は、管理者、エージェント、およびエンドエンティティーにより取り消されます。エージェント権限を持つエージェントおよび管理者は、エージェントサービスページでフォームを使用して証明書を取り消すことができます。エンドユーザーは、エンドエンティティーインターフェイスの **失効** タブのフォームを使用して証明書を失効させることができます。エンドユーザーは自分の証明書のみを取り消すことができますが、エージェントと管理者はサーバーによって発行された証明書を取り消すことができます。証明書を取り消すには、エンドユーザーもサーバーに対する認証に必要になります。

証明書が取り消されると、Certificate Manager は内部データベースの証明書のステータスを更新します。サーバーは、内部データベースのエントリーを使用して、取り消されたすべての証明書を追跡します。設定すると、CRL を中央リポジトリーに公開して公開し、リスト内の証明書が無効になったことを他のユーザーに通知します。

7.1.3. CRL 発行ポイント

CRL は非常に大きくなる可能性があるため、大きな CRL の取得と配信のオーバーヘッドを最小限に抑える方法は複数あります。この方法の1つは、証明書領域全体を分割し、個別の CRL を各パーティションに関連付けます。このパーティションは *CRL 発行ポイント* と呼ばれます。これは、失効した全証明書のサブセットが保持される場所です。パーティション設定は、取り消された証明書が CA 証明書であるかどうか、特定の理由で取り消されたかどうか、または特定のプロファイルを使用して発行されたかどうかに基づいて行うことができます。各発行ポイントは名前で識別されます。

デフォルトでは、Certificate Manager は単一の CRL (マスター *CRL*) を生成し、公開します。発行ポイントは、すべての証明書、CA 署名証明書のみ、または期限切れの証明書を含むすべての証明書の CRL を生成できます。

発行ポイントを定義したら、それらを証明書に含めることができるため、証明書の失効ステータスを確認する必要があるアプリケーションは、マスターまたはメイン CRL の代わりに、証明書で指定された CRL 発行ポイントにアクセスできます。発行ポイントで維持される CRL はマスター CRL よりも小さいため、失効ステータスの確認ははるかに高速です。

CRL ディストリビューションポイントは、**GRLDistributionPoint** 拡張機能を設定して証明書に関連付けることができます。

7.1.4. Delta CRL

デルタ CRL は、定義された発行ポイントに対して発行できます。デルタ CRL には、完全な CRL への最後の更新以降に取り消された証明書に関する情報が含まれます。発行先の Delta CRL は、**DeltaCRLIndicator** 拡張を有効にして作成されます。

7.1.5. CRL の公開

Certificate Manager は CRL をファイル、LDAP 準拠のディレクトリー、または OCSP レスポンダーに公開できます。CRL が公開される場所と頻度は、*9章 証明書および CRL の公開* で説明されているように、証明書マネージャーで設定されます。

CRL は非常に大きくなる可能性があるため、CRL の公開には非常に長い時間がかかる可能性があり、プロセスが中断される可能性があります。特別なパブリッシャーは、HTTP 1.1 を介して CRL をファイルに発行するように設定できます。プロセスが中断された場合、CA サブシステムの Web サーバーは、最初からではなく、中断された時点から発行を再開できます。この操作は、「再開可能な CRL ダウンロードの設定」 に説明があります。

7.1.6. 証明書失効ページ

Certificate Manager のエンドエンティティーページには、SSL/TLS クライアントによって認証されたデフォルトの HTML フォームが含まれます。フォームは **Revocation** タブからアクセスできます。**User Certificate** リンクをクリックすると、このような失効のフォームが表示されます。

組織の要件に合わせてフォームの外観を変更するには、**UserRevocation.html** (SSL/TSL クライアント認証によるクライアントまたは個人証明書の失効を許可するフォーム) を編集します。ファイルは **/var/lib/*instance_name*/webapps/*subsystem_type*/ee/*subsystem_type*** ディレクトリーにあります。

7.2. CMC 失効の実行

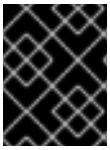
CMS (CMC) 登録を介した Certificate Management と同様に、CMC 失効により、ユーザーは失効クライアントをセットアップし、一致する **subjectDN** 属性を使用するエージェント証明書またはユーザー証明書のいずれかを使用して失効要求に署名できます。これにより、ユーザーは署名済み要求を Certificate Manager に送信できます。

または、Shared Secret Token メカニズムを使用して CMC の失効を認証することもできます。詳細は、『CMC 共有シークレット機能の有効化』 を参照してください。

ユーザーまたはエージェントが要求に署名するかどうか、または Shared Secret Token が使用されているかどうかに関係なく、Certificate Manager は、有効な失効要求を受信すると、証明書を自動的に失効させます。

Certificate System は、CMC 失効要求用に以下のユーティリティーを提供します。

- **CMCRequest**詳細は、「**CMCRequest** を使用した証明書の取り消し」 を参照してください。
- **CMCRevoke**詳細は、「**CMCRevoke**を使用した証明書の取り消し」 を参照してください。



重要

Red Hat は、**CMCRequest** ユーティリティーを使用して、失効要求の生成を推奨します。これは、**CMCRevoke** よりも多くのオプションを提供するためです。

7.2.1. CMCRequest を使用した証明書の取り消し

CMCRequest を使用して証明書を取り消すには、以下を実行します。

1. 以下の内容で、**/home/user_name/cmc-request.cfg** などの CMC 失効要求の設定ファイルを作成します。

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1

#output: full path for the CMC request in binary format
output=/home/user_name/cmc.revoke.userSigned.req

#tokenname: name of token where user signing cert can be found
#(default is internal)
tokenname=internal

#nickname: nickname for user signing certificate which will be used
#to sign the CMC full request.
nickname=signer_user_certificate

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
dbdir=/home/user_name/.dogtag/nssdb/

#password: password for cert9.db which stores the user signing
#certificate and keys
password=myPass

#format: request format, either pkcs10 or crmf.
format=pkcs10

## revocation parameters
revRequest.enable=true
revRequest.serial=45
revRequest.reason=unspecified
revRequest.comment=user test revocation
revRequest.issuer=issuer
revRequest.sharedSecret=shared_secret
```

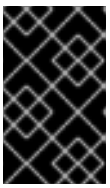
2. CMC 要求を作成します。

```
# CMCRequest /home/user_name/cmc-request.cfg
```

コマンドが成功すると、**CMCRequest** ユーティリティーは、要求設定ファイルの ***output*** パラメーターで指定されたファイルに CMC 要求を保存します。

3. **/home/user_name/cmc-submit.cfg** などの設定ファイルを作成します。このファイルは、後で CMC 取り消し要求を CA に送信します。作成されたファイルに以下の内容を追加します。

```
#host: host name for the http server
host=>server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be
#in binary format
input=/home/user_name/cmc.revoke.userSigned.req

#output: full path for the response in binary format
output=/home/user_name/cmc.revoke.userSigned.resp

#tokenname: name of token where SSL client authentication certificate
#can be found (default is internal)
#This parameter will be ignored if secure=false
tokenname=internal

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
#This parameter will be ignored if secure=false
dbdir=/home/user_name/.dogtag/nssdb/

#clientmode: true for client authentication, false for no client
#authentication. This parameter will be ignored if secure=false
clientmode=true

#password: password for cert9.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=signer_user_certificate
```



重要

CMC 失効要求に署名されている場合は、***secure*** パラメーターおよび ***clientmode*** パラメーターも **true** に設定します。さらに ***nickname*** パラメーターも入力します。

4. 要求に署名したユーザーに応じて、**HttpClient** の設定ファイルの ***servlet*** パラメーターを適切に設定する必要があります。

- エージェントが要求に署名した場合は、以下を設定します。

```
servlet=/ca/ee/ca/profileSubmitCMCFull
```

- ユーザーが要求に署名した場合は、以下を設定します。

```
servlet=/ca/ee/ca/profileSubmitSelfSignedCMCFull
```

5. CMC 要求を送信します。

```
# HttpClient /home/user_name/cmc-submit.cfg
```

CMCRequest を使用して証明書の取り消しの詳細は、CMCRequest(1) man ページを参照してください。

7.2.2. CMCRevokeを使用した証明書の取り消し

CMC 失効ユーティリティー **CMCRevoke** は、エージェントの証明書を使用して失効要求に署名するために使用されます。このユーティリティーは、必要な情報 (証明書のシリアル番号、発行者名、失効理由) を渡して取り消す証明書を識別し、次に失効を実行する CA エージェントを識別するための必要な情報 (証明書のニックネームと証明書のあるデータベース) を渡します。

証明書が取り消される理由は、次のいずれかです (数値は、**CMCRevoke** に渡される値です)。

- **0**: 指定無し
- **1**: キーが侵害されました。
- **2**: CA キーが侵害されました。
- **3**: 従業員の所属が変更になりました
- **4**: 証明書が置き換えられました
- **5**: 運用停止
- **6**: 証明書が保留中です

利用可能なツール引数は、『コマンドラインツールツールガイド』 で詳細に説明されています。

7.2.2.1. CMCRevoke のテスト

1. 既存の証明書の CMC 失効要求を作成します。

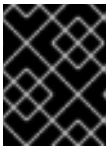
```
CMCRevoke -d/path/to/agent-cert-db -nnickname -iissuerName -sserialName -mreason -
ccomment
```

たとえば、エージェント証明書を含むディレクトリーは **~jsmith/.mozilla/firefox/** で、証明書のニックネームは **AgentCert** で、証明書のシリアル番号は **22** の場合、コマンドは次のとおりです

```
CMCRevoke -d"~jsmith/.mozilla/firefox/" -n"ManagerAgentCert" -i"cn=agentAuthMgr" -s22 -
m0 -c"test comment"
```

注記

引用符で囲まれたスペースを含む値を囲みます。



重要

引数とその値の間には空白を入れないでください。たとえば、26 のシリアル番号は **-26** ではなく、**-s 26** となります。

2. エンドエンティティーを開きます。

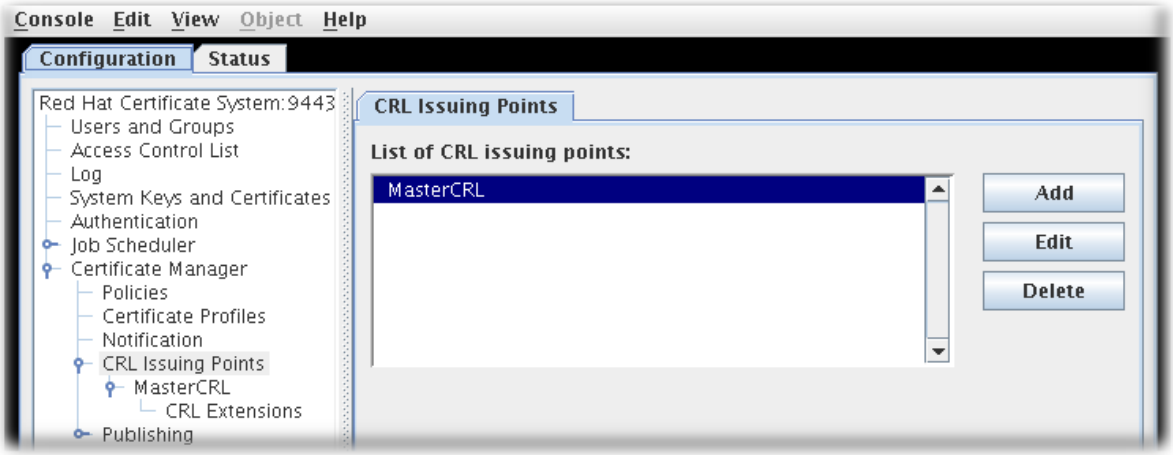
```
https://server.example.com:8443/ca/ee/ca
```

3. **Revocation** タブを選択します。
4. メニューの **CMC Revoke** リンクを選択します。
5. **CMCRevoke** からテキストエリアに出力を貼り付けます。
6. 貼り付けられたコンテンツから **-----BEGIN NEW CERTIFICATE REQUEST-----** および **----END NEW CERTIFICATE REQUEST-----** を削除します。
7. **送信** をクリックします。
8. 返されるページは、正しい証明書が取り消されていることを確認します。

7.3. CRL の実行

1. Certificate Manager は、その CA 署名証明書キーを使用して CRL に署名します。CRL に個別の署名キーペアを使用するには、CRL 署名キーを設定し、このキーを使用して CRL に署名するように Certificate Manager の設定を変更します。詳細は、「異なる証明書を使用するように CA を設定して CRL を署名」を参照してください。
2. CRL 発行ポイントの設定発行ポイントは、マスター CRL に対してすでにセットアップされ、有効にされています。

図7.1 デフォルトの CRL 発行ポイント



CRL の追加の発行ポイントを作成できます。詳細は、「発行ポイントの設定」を参照してください。

発行ポイントを設定して CRL のリストを定義するときに設定したオプションに応じて、発行ポイントが作成できる CRL には 5 つのタイプがあります。

- **マスター CRL** には、CA 全体から失効した証明書の一覧が含まれます。

- **ARL** は、失効した CA 証明書のみが含まれる Authority Revocation List です。
- **期限切れの証明書を持つ CRL** には、CRL で有効期限が切れた証明書が含まれます。
- **証明書プロファイルの CRL** は、最初に証明書を作成するために使用されるプロファイルに基づいて、失効した証明書を判別します。
- **理由コードによる CRL** は、失効した理由コードに基づいて、失効した証明書を判別します。

3. 各発行ポイントに CRL を設定します。詳細は、「各発行ポイントの CRL の設定」を参照してください。
4. 発行ポイントに設定された CRL 拡張機能を設定します。詳細は、「CRL 拡張機能の設定」を参照してください。
5. 発行ポイントの拡張を有効にすることにより、発行ポイントにデルタ CRL を設定するか、または発行ポイント **DeltaCRLIndicator** または **CRLNumber** の拡張を有効にします。
6. 発行先に関する情報が含まれるように **CRLDistributionPoint** 拡張機能を設定します。
7. ファイル、LDAP ディレクトリー、または OCSP レスポンダーへの公開 CRL を設定します。公開の設定の詳細は、*9章 証明書および CRL の公開* を参照してください。

7.3.1. 発行ポイントの設定

発行ポイントは、新しい CRL に含まれる証明書を定義します。マスター CRL 発行ポイントは、Certificate Manager の失効した証明書の一覧を含むマスター CRL 用にデフォルトで作成されます。

新規の発行ポイントを作成するには、以下の手順を実施します。

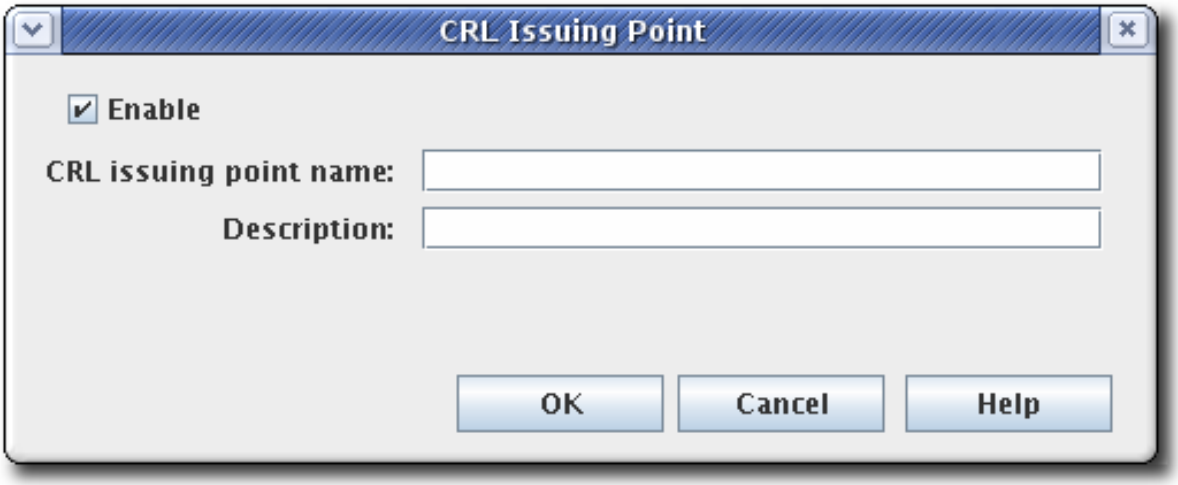
1. 証明書システムコンソールの起動

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションメニューから **Certificate Manager** を展開します。次に、**CRL Issuing Points** を選択します。
3. 発行ポイントを編集するには、発行ポイントを選択して、**Edit** をクリックします。編集できるパラメーターは、発行ポイントの名前と、発行ポイントが有効か無効かだけです。

   発行ポイントを追加するには、**Add** をクリックします。CRL Issuing Point エディターウインドウが開きます。

図7.2 CRL Issuing Point エディター



注記

一部のフィールドがコンテンツを読み取るのに十分な大きさで表示されない場合は、コーナーの1つをドラッグしてウィンドウを拡大します。

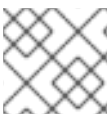
以下のフィールドに入力します。

- **Enable**。選択した場合は発行ポイントを有効にします。無効にする場合は選択を解除します。
- **CRL Issuing Point name**。発行ポイントの名前を指定します。スペースは使用できません。
- **Description**。発行ポイントを説明します。

4. **OK** をクリックします。

新しい発行ポイントを表示して設定するには、CA コンソールを閉じ、その後にコンソールを再度開きます。新しい発行ポイントは、ナビゲーションツリーの **CRL Issuing Points** エントリーの下に一覧表示されます。

新しい発行ポイントに CRL を設定し、CRL と使用する CRL 拡張機能を設定します。発行ポイントの設定に関する詳細は、「各発行ポイントの CRL の設定」 を参照してください。CRL 拡張機能の設定に関する詳細は、「CRL 拡張機能の設定」 を参照してください。作成された CRL はすべて、エージェントサービスページの **Update Revocation List** ページに表示されます。



注記

pkiconsole が非推奨になりました。

7.3.2. 各発行ポイントの CRL の設定

生成間隔、CRL バージョン、CRL 拡張、署名アルゴリズムなどの情報はすべて、発行ポイントの CRL 用に設定できます。CRL は発行ポイントごとに設定する必要があります。

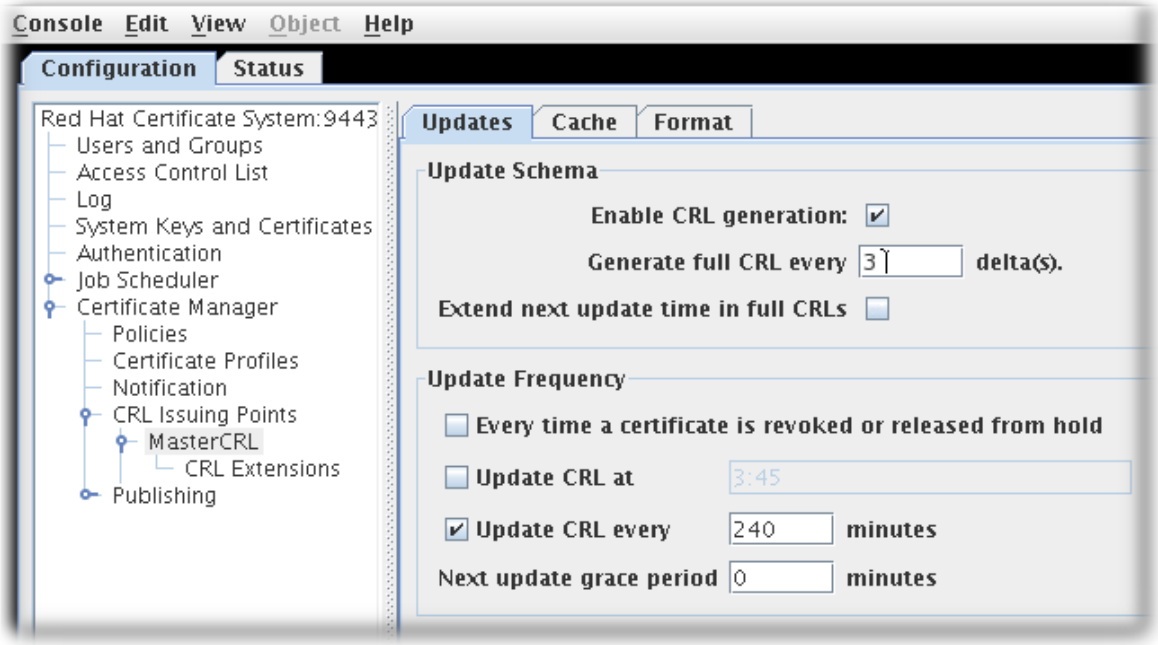
1. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. ナビゲーションツリーで、**Certificate Manager** を選択し、**CRL Issuing Points** を選択します。

3. **Issuing Points** エントリーの下に、発行ポイント名を選択します。

4. 発行ポイントの **Update** タブに情報を指定して、CRL の更新方法および頻度を設定します。このタブには、**Update Schema** および **Update Frequency** の 2 つのセクションがあります。



- **Update Schema** セクションには以下のオプションが含まれます。
  - **CRL 生成を有効にします**。このチェックボックスは、発行ポイントに CRL が生成されるかどうかを設定します。
  - **Generate full CRL every # delta(s)**。このフィールドは、変更の数に関連して CRL が作成された頻度を設定します。
  - **Extend next update time in full CRLs**。これにより、生成された CRL に **nextUpdate** フィールドを設定するオプションが提供されます。***nextUpdate*** パラメーターは、フル CRL かデルタ CRL かに関係なく、次の CRL が発行される日付を示します。フル CRL とデルタ CRL の組み合わせを使用している場合は、***Extend next update time in full CRLs*** を有効にすると、フル CRL の ***nextUpdate*** パラメーターに次の ***フル*** CRL が発行されるタイミングを表示させることができます。それ以外の場合は、フル CRL の ***nextUpdate*** パラメーターは、そのデルタが次に発行される CR L になるため、次の ***デルタ*** CRL がいつ発行されるかを示します。
- **Update Frequency** セクションは、CRL が生成され、ディレクトリーに発行されたときに異なる間隔を設定します。
  - **Every time a certificate is revoked or released from hold**。これにより、証明書を取り消すたびに Certificate Manager が CRL を生成するよう設定されます。Certificate Manager は、CRL が生成されるたびに、設定されたディレクトリーに CRL を発行しようとします。CRL の生成は、CRL のサイズが大きい場合に消費できます。証明書が取り消されるたびに CRL を生成するように Certificate Manager を設定すると、サーバーがかなりの時間使用される可能性があります。この間、サーバーは受け取った変更でディレクトリーを更新できなくなります。

この設定は、標準的なインストールには推奨されません。このオプションは、サーバーが CRL をフラットファイルに発行したかどうかのテストなど、すぐに失効をテストするために選択する必要があります。

- **Update the CRL at**。このフィールドは、CRL を更新する必要がある毎日の時間を設定します。複数回指定するには、**01:50,04:55,06:55** などのコンマ区切りリストを入力します。複数日のスケジュールを入力するには、コンマ区切りのリストを入力して同じ日の時間を設定し、セミコロンで区切ったリストを入力して異なる日の時間を識別します。たとえば、これは、サイクルの 1 日目の 午前 1:50、4:55、および 6:55、そして 2 日目の午前 2:00、5:00、および午後 5:00 に失効を設定します。

  ```
  01:50,04:55,06:55;02:00,05:00,17:00
  ```

- **CRL をすべて更新** します。このチェックボックスでは、フィールドに設定された間隔で CRL を生成できます。たとえば、毎日 CRL を発行するには、チェックボックスを選択して、このフィールドに **1440** を入力します。

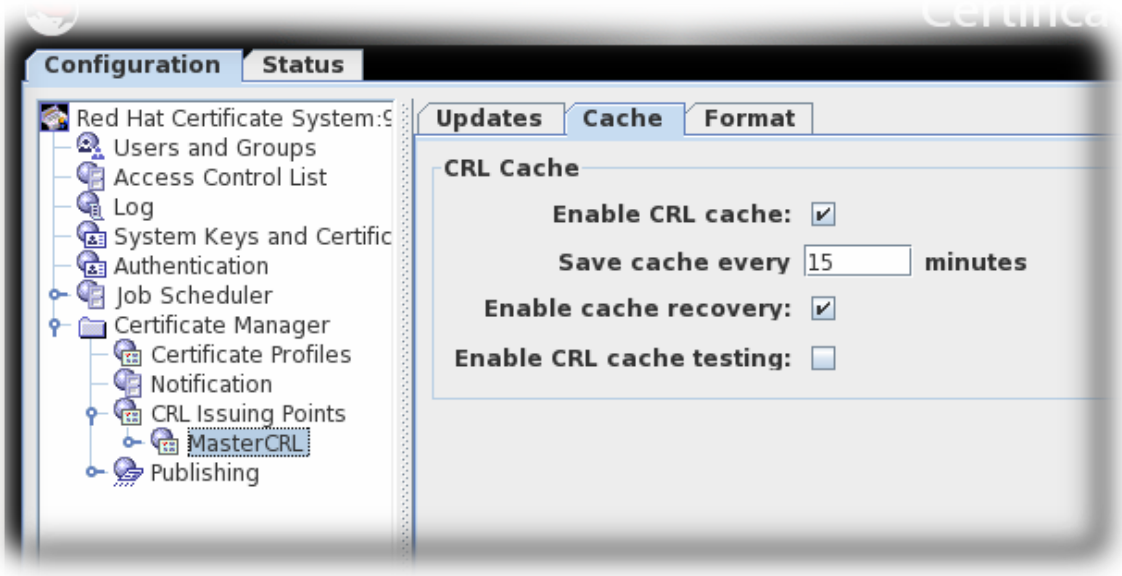
- **Next update grace period**。Certificate Manager が特定の頻度で CRL を更新する場合、サーバーは、CRL を作成して発行する時間を確保するために、次の更新時間までの猶予期間を持つように設定できます。たとえば、サーバーが 2 分の猶予期間で 20 分ごとに CRL を更新するように設定されていて、CRL が 16:00 に更新された場合、CRL は 16:18 に再更新されます。

重要

既知の問題により、現在フルおよびデルタの証明書失効リストのスケジュールを設定している場合、**Update CRL every time a certificate is revoked or released from hold** オプションでは、2 つの **grace period** 設定を記入する必要もあります。したがって、このオプションを選択するには、最初に **Update CRL every** オプションを選択して、する必要がありますし、**Next update grace period # minutes** ボックスに番号を入力する必要があります。

5. **Cache** タブは、キャッシュが有効であるかどうかとキャッシュ頻度を設定します。

図7.3 CRL キャッシュタブ



- **Enable CRL cache**。このチェックボックスは、デルタ CRL の作成に使用されるキャッシュを有効にします。キャッシュが無効になっている場合は、デルタ CRL は作成されません。キャッシュの詳細は、「証明書の失効について」 を参照してください。

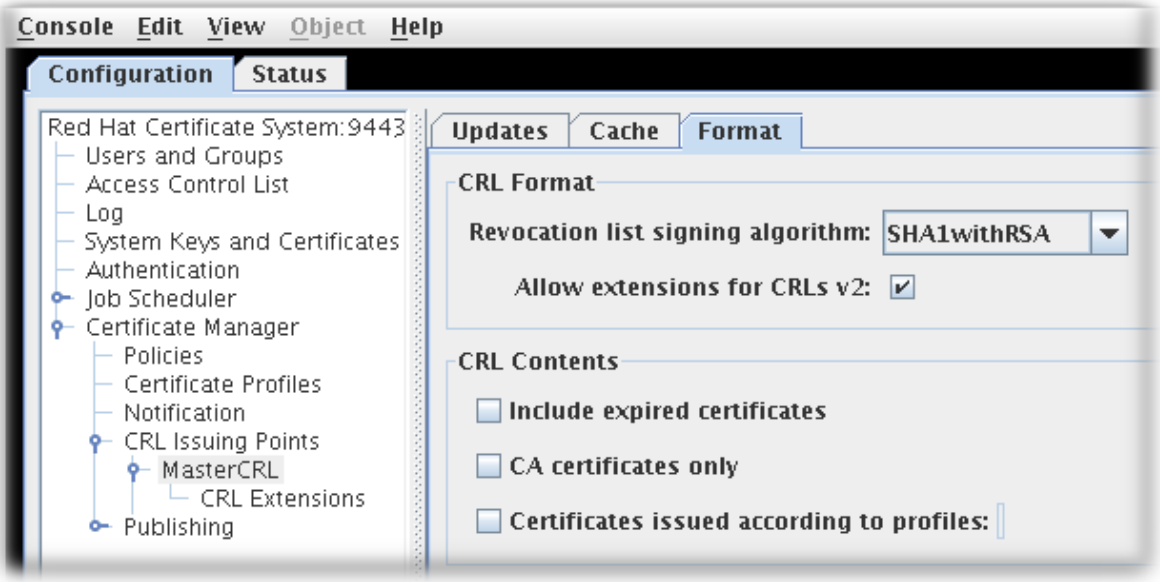
- **キャッシュを毎回更新** します。このフィールドは、キャッシュが内部データベースに書き込む頻度を設定します。証明書が取り消されるたびに、キャッシュをデータベースに書き出すには、**0** に設定します。

- **キャッシュリカバリーを有効** にします。このチェックボックスを選択すると、キャッシュを復元できます。

- **Enable CRL cache testing**。このチェックボックスは、特定の CRL 発行ポイントの CRL パフォーマンステストを有効にします。このオプションで生成された CRL は、デプロイした CA では使用しないでください。テスト目的で発行された CRL には、パフォーマンステストのみを目的として生成されたデータが含まれているためです。

6. **フォーマット** タブでは、作成される CRL のフォーマットおよびコンテンツを設定します。**CRL Format** および **CRL Contents** の 2 つのセクションがあります。

図7.4 CRL 形式タブ



- **CRL Format** セクションには、以下の 2 つのオプションがあります。

  - **Revocation list signing algorithm** は、CRL 暗号化を行うために許可された暗号のドロップダウンの一覧です。

  - **Allow extensions for CRL v2** するには、発行ポイントに CRL v2 拡張を有効にするチェックボックスがあります。これが有効な場合は、「CRL 拡張機能の設定」 で説明されている必要な CRL 拡張機能を設定します。

注記

CRL を作成するには、拡張機能を有効にする必要があります。

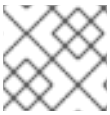
- **CRL Contents** セクションには、CRL に追加する証明書のタイプを設定する 3 つのチェックボックスがあります。

  - **期限切れの証明書** を含めます。これには、期限切れになった証明書が含まれます。これを有効にすると、失効した証明書に関する情報は、証明書の期限が切れた後も CRL

に残ります。これが有効になっていないと、証明書の有効期限が切れると、失効した証明書に関する情報が削除されます。

- **CA 証明書のみ**これには、CRL の CA 証明書のみが含まれます。このオプションを選択すると、失効した CA 証明書のみを一覧表示する Authority Revocation List (ARL) が作成されます。
- **プロファイルに従って発行された証明書**。これには、リストされたプロファイルに従って発行された証明書のみが含まれます。複数のプロファイルを指定するには、コンマ区切りのリストを入力します。

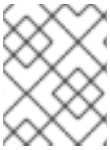
7. **Save** をクリックします。
8. この発行ポイントでは、拡張機能は可能で、設定できます。詳細は、「CRL 拡張機能の設定」を参照してください。



注記

pkiconsole が非推奨になりました。

7.3.3. CRL 拡張機能の設定



注記

拡張機能には、発行ポイントに CRLs v2 の **Allow extensions for CRLs v2** チェックボックスが選択されている場合にのみ、発行ポイントに必要です。

発行ポイントが作成されると、3 つの拡張機能 (**CRLReason**、**InvalidityDate**、および **CRLNumber**) が自動的に有効になります。その他の拡張は利用できますが、デフォルトで無効になっています。これは、有効化および変更できます。利用可能な CRL 拡張の詳細は、??? を参照してください。

CRL 拡張機能を設定するには、以下を行います。

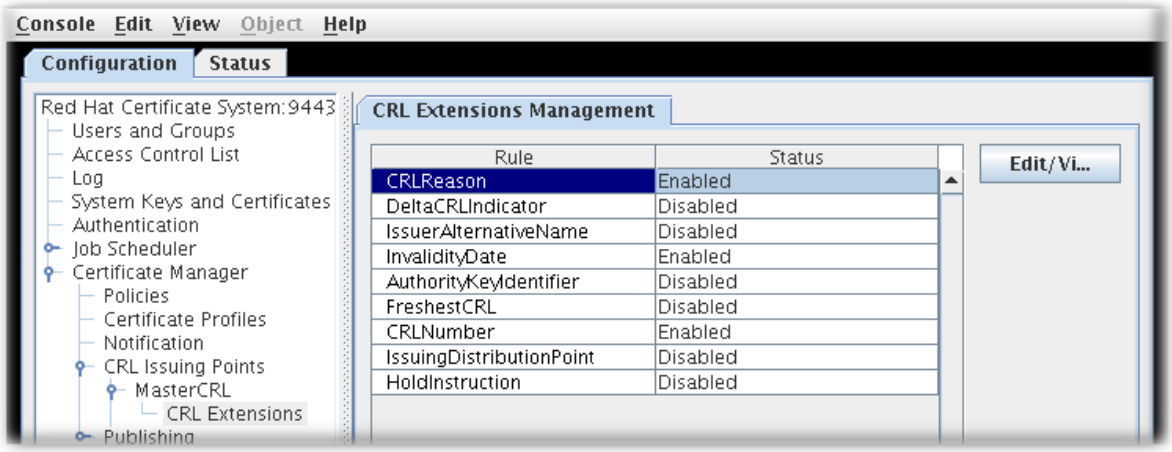
1. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. ナビゲーションツリーで、**Certificate Manager** を選択し、**CRL Issuing Points** を選択します。
3. **Issuing Points** エントリーの下にある発行ポイント名を選択し、発行ポイントの下にある **CRL 拡張** エントリーを選択します。

   右側のペインには、設定された拡張機能を一覧表示する **CRL Extensions Management** タブが表示されます。

図7.5 CRL 拡張機能



4. ルールを変更するには、ルールを選択し、**Edit/View** をクリックします。

5. ほとんどの拡張には 2 つのオプションがあり、有効にして、重要なかどうかを設定します。詳細情報が必要なものもあります。必要な値をすべて指定します。各拡張機能およびそれらの拡張機能のパラメーターに関する詳細は、??? を参照してください。

6. **OK** をクリックします。

7. **Refresh** をクリックし、すべてのルールの更新されたステータスを表示します。

注記

pkiconsole が非推奨になりました。

7.3.4. 異なる証明書を使用するように CA を設定して CRL を署名

CS.cfg ファイルを編集してこの機能を設定する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『異なる証明書を使用するように CA を設定して CRL を署名』 セクションを参照してください。

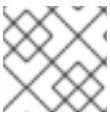
7.3.5. キャッシュからの CRL の生成

デフォルトでは、CRL は CA の内部データベースから生成されます。ただし、証明書が取り消されてメモリーに保持されるため、失効情報を収集できます。その後、この失効情報を使用して、メモリーから CRL を更新できます。内部データベースから CRL を生成するために必要なデータベース検索を省略すると、パフォーマンスが大幅に改善されます。

注記

キャッシュから CRL を生成する際のパフォーマンスの向上により、ほとんどの環境で ***enableCRLCache*** パラメーターが有効になります。ただし、実稼働環境ではこの ***Enable CRL cache testing*** パラメーターを有効に **しないでください**。

7.3.5.1. コンソールでのキャッシュからの CRL 生成の設定



注記

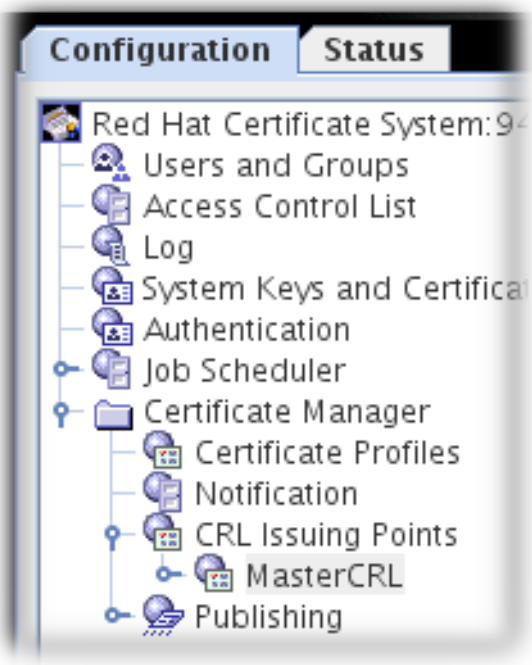
pkiconsole が非推奨になりました。

1. コンソールを開きます。

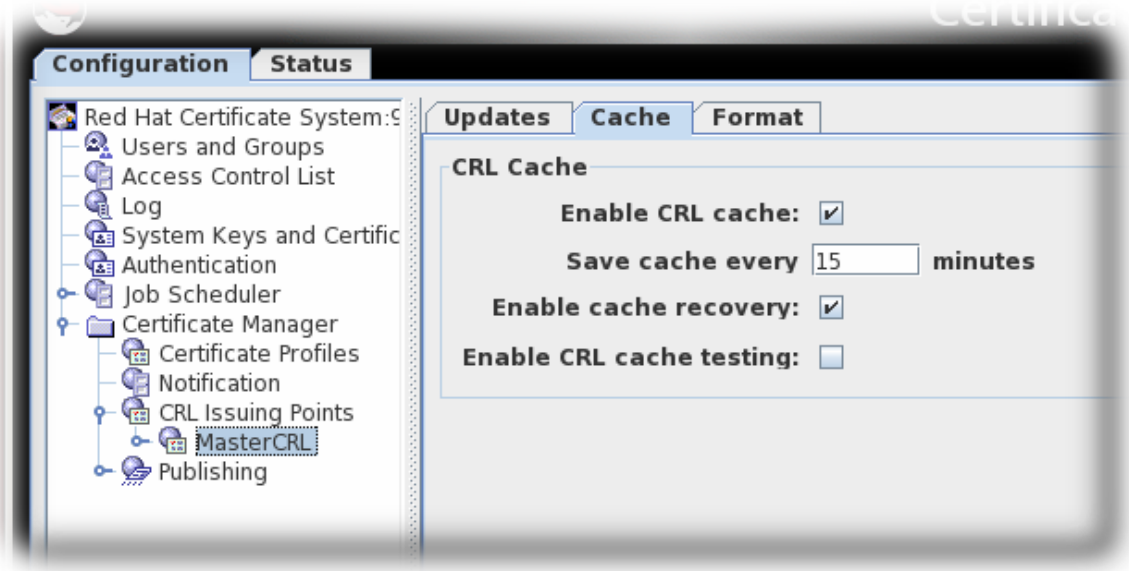
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、**Certificate Manager** フォルダーと **CRL Issuing Points** サブディレクトリーを展開します。

3. **MasterCRL** ノードを選択します。



4. **Enable CRL cache** を選択します。



5. 変更を保存します。

7.3.5.2. CS.cfg のキャッシュからの CRL 生成の設定

CS.cfg ファイルを編集してこの機能を設定する方法は、『Red Hat Certificate System の計画、インストール、およびデプロイメントのガイド』 の 『CS.cfg のキャッシュからの CRL 生成の設定』 セクションを参照してください。

7.4. FULL および DELTA CRL スケジュールの設定

CRL は定期的に生成されます。「各発行ポイントの CRL の設定」 の設定でその期間は切り替わるものです。

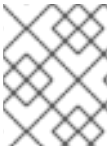
CRL は、時間ベースのスケジュールに従って発行されます。CRL は、証明書が失効するたびに、1 日の特定の時間帯に、または数十分に 1 回発行することができます。

時間ベースの CRL 生成スケジュールは、生成されるすべての CRL に適用されます。CRL には完全な CRL とデルタ CRL の 2 つの種類があります。完全な CRL には、取り消されたすべての証明書のレコードがありますが、デルタ CRL には、最後の CRL (デルタまたは完全) が生成されてから取り消された証明書のみが含まれます。

デフォルトでは、完全な CRL はスケジュールで指定した間隔で生成されます。正確な **delta** CRL を生成することで、完全な CRL を生成するまでに時間がかかる場合があります。生成間隔は **CRL スキーマ** で設定され、デルタと完全な CRL を生成するスキームを設定します。

たとえば、間隔が 3 に設定されている場合、生成される最初の CRL はフル CRL とデルタ CRL の両方になり、次の 2 つの世代の更新はデルタ CRL のみになり、4 番目の間隔は再びフル CRL とデルタ CRL の両方になります。つまり、3 番目の生成間隔はすべて完全な CRL とデルタ CRL の両方があります。

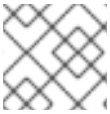
```
Interval   1, 2, 3, 4, 5, 6, 7 ...
Full CRL   1        4        7 ...
Delta CRL  1, 2, 3, 4, 5, 6, 7 ...
```



注記

完全な CRL に加えてデルタ CRL を生成するには、CRL キャッシュを有効にする必要があります。

7.4.1. コンソールでの CRL 更新間隔の設定



注記

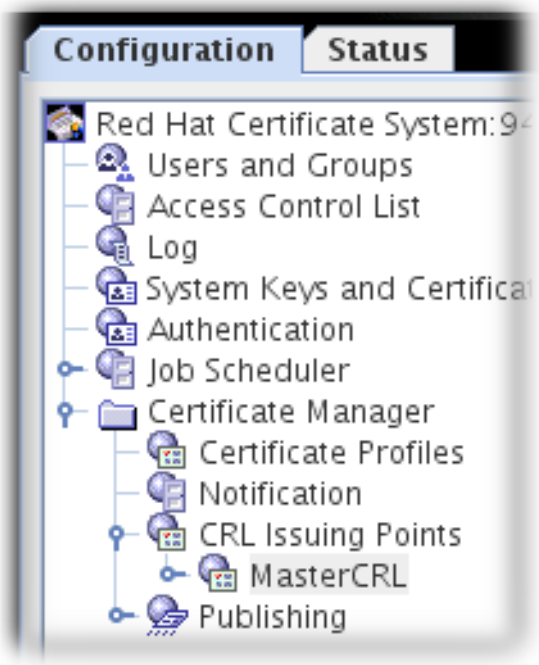
pkiconsole が非推奨になりました。

1. コンソールを開きます。

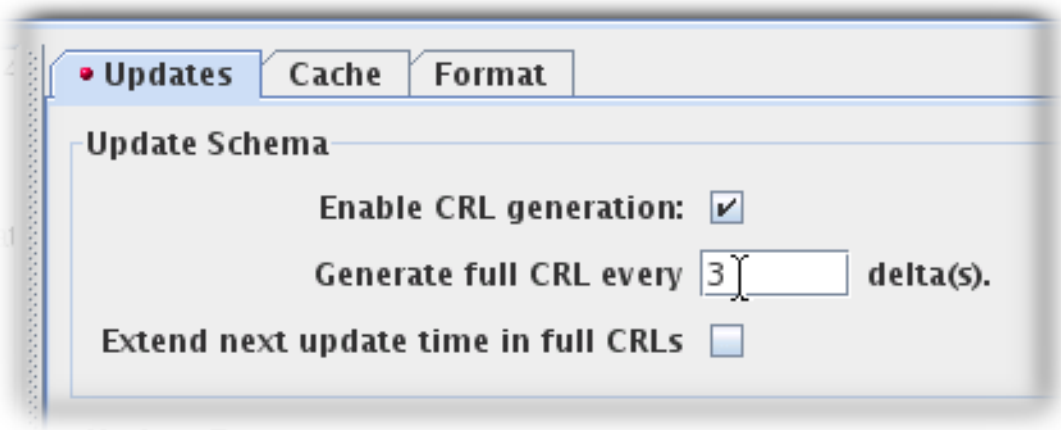
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、**Certificate Manager** フォルダーと **CRL Issuing Points** サブディレクトリーを展開します。

3. **MasterCRL** ノードを選択します。



4. **Generate full CRL every # delta(s)** フィールドに、必要な間隔を入力します。

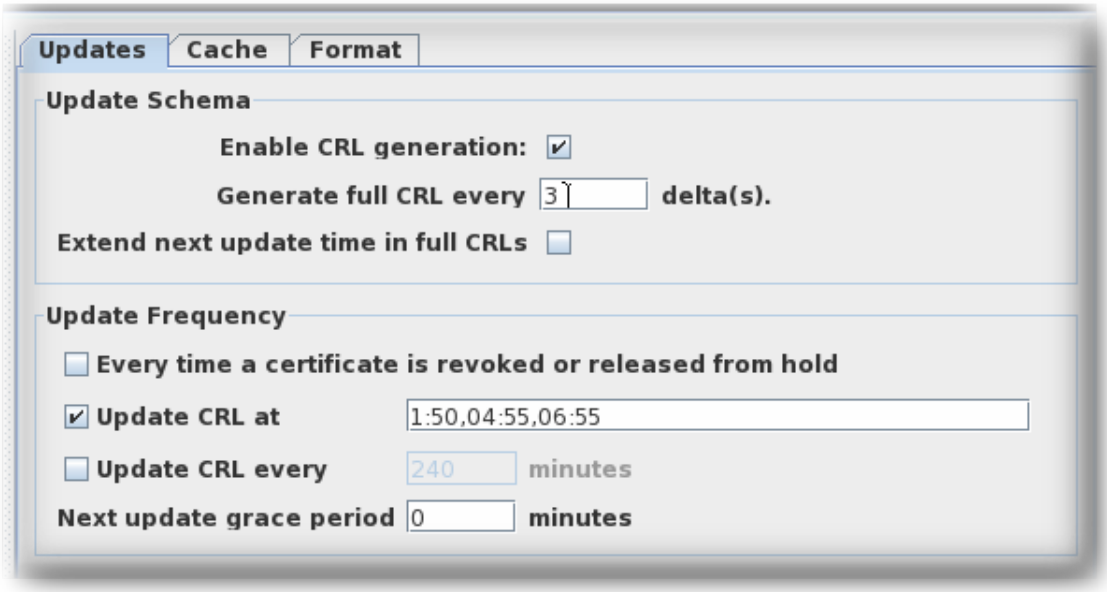


5. 証明書失効の機会、周期的な間隔、または更新が発生する時間を設定することにより、更新頻度を設定します。

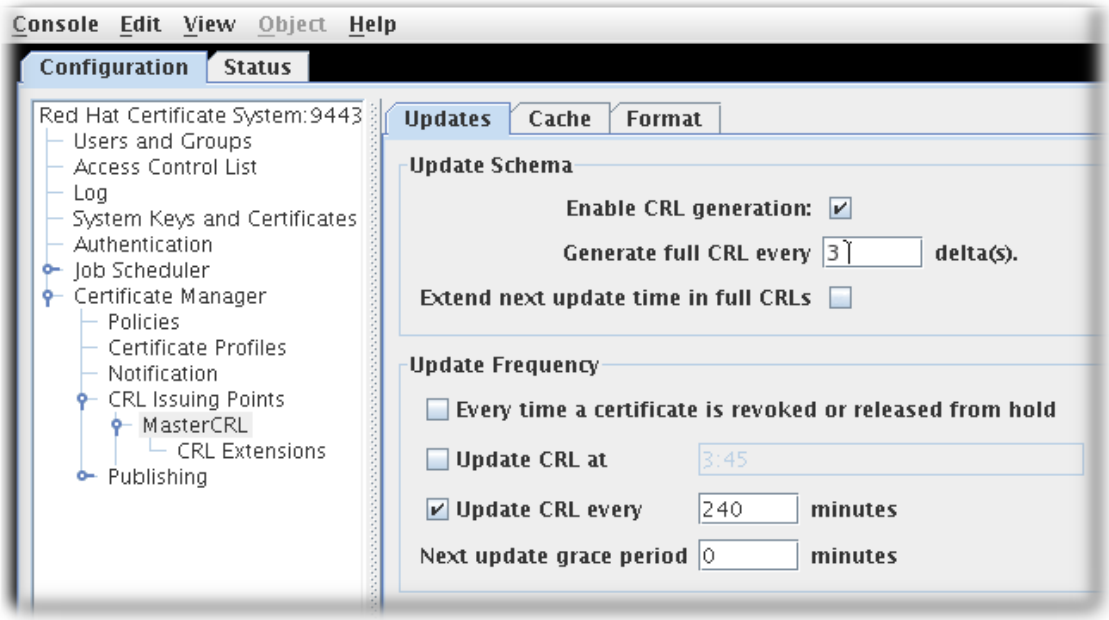
    - **Update CRL every time a certificate is revoked or released from hold** チェックボックスを選択します。**Update CRL every time a certificate is revoked or released from hold** オプションでも、2 つの **Grace period** 設定を入力する必要があります。これは既知の問題で、バグは Red Hat Bugzilla で追跡されています。

    - **Update CRL every time a certificate is revoked or released from hold** チェックボックスを選択します。

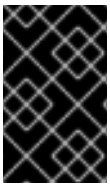
    - **Update CRL at** チェックボックスを選択し、**01:50,04:55,06:55** などの特定の時刻をコンマで区切って入力します。



- **Update CRL every** チェックボックスを選択し、**240** などの必要な間隔を入力します。



6. 変更を保存します。



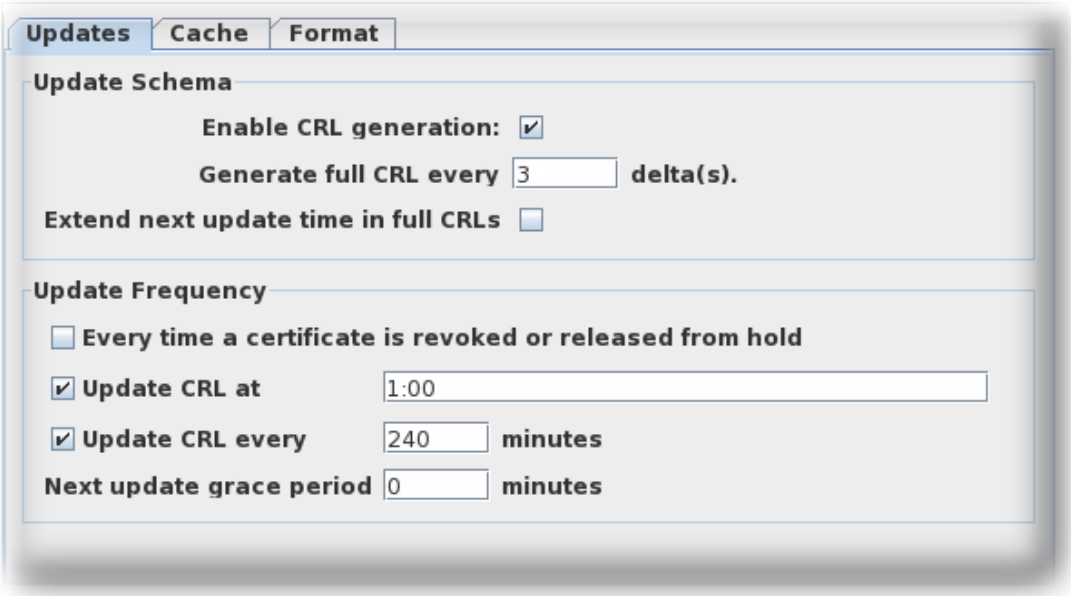
重要

Update CRL every time a certificate is revoked or released from hold オプションでも、2 つの **grace period** 設定を入力する必要があります。これは既知の問題で、バグは Red Hat Bugzilla で追跡されています。

注記

間隔ごとに CRL を更新するとドリフトが発生する場合があります。通常、ドリフトは手動更新と CA の再起動時に実行されます。

スケジュールのずれを防ぐには、**Update CRL at** チェックボックスを選択して値を入力します。間隔の更新は、24 時間ごとに **Update CRL at** 値と再同期します。



間隔で CRL を更新する場合は、**Update CRL at** 値は 1 つだけ受け入れられます。

7.4.2. CS.cfg での CRL の更新間隔の設定

CS.cfg ファイルを編集してこの機能を設定する方法は、『Red Hat Certificate System の計画、インストール、およびデプロイメントのガイド』 の 『CS.cfg での CRL の更新間隔の設定』 セクションを参照してください。

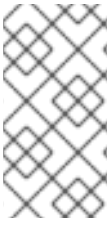
7.4.3. 複数の日における CRL 生成スケジュールの設定

デフォルトで、CRL 生成のスケジュールは 24 時間に対応しています。また、デフォルトでは、フル CRL とデルタ CRL が有効になっている場合、1 つまたはすべてのデルタ CRL の代わりに、特定の間隔、つまり 3 回の更新ごとにフル CRL が発生します。

複数日にわたる CRL 生成スケジュールを設定するには、時間のリストでコンマを使用して同じ日の時間を区切り、セミコロンを使用して日を区切ります。

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00;02:00,05:00,17:00
```

この例では、スケジュールの 1 日目の 01:00、03:00、および 18:00 と、スケジュールの 2 日目の 02:00、05:00、および 17:00 に CRL を更新します。3 日目にサイクルが再開します。



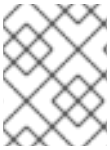
注記

セミコロンは新規日を示します。セミコロンで一覧を開始すると、CRL が生成されない最初の日になります。同様に、リストをセミコロンで終了すると、CRL が生成されないスケジュールに最終日が追加されます。2 つのセミコロンを合わせると、CRL が生成されない日になります。

デルタ更新とは独立してフル CRL 更新を設定するために、時間のリストは、完全な CRL 更新がいつ発生するかを示すアスタリスクが前に付いた時間値を受け入れます。

```
ca.crl.MasterCRL.dailyUpdates=01:00,03:00,18:00,*23:00;02:00,05:00,21:00,*23:30
```

この例では、1 日目の 01:00、03:00、および 18:00 にデルタ CRL 更新を生成し、23:00 にフル CRL およびデルタ CRL の更新を生成します。2 日目では、デルタ CRL は 02:00、05:00、および 21:00 で更新されます。これは、フル CRL およびデルタ CRL の更新が 23:30 で行われます。3 日目にサイクルが再開します。



注記

セミコロンとアスタリスク構文は、コンソールと **CS.cfg** ファイルを手動で編集する時に機能します。

7.5. 失効チェックの有効化

失効チェック とは、Certificate System サブシステムが、エージェントまたは管理者がインスタンスの安全なインターフェイスにアクセスしようとしたときに、証明書が有効であり、失効していないことを確認することを意味します。これは、ローカルの OCSP サービス (CA の内部 OCSP サービスまたは個別の OCSP レスポンダー) を使用して証明書の失効ステータスを確認します。

OCSP 設定は、「OCSP (Online Certificate Status Protocol) レスポンダーの使用」 で説明されています。

『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『CA での自動失効チェックの有効化』を参照してください。

『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『サブシステムの証明書失効チェックの有効化』を参照してください。

7.6. OCSP (ONLINE CERTIFICATE STATUS PROTOCOL) レスポンダーの使用

7.6.1. OCSP レスポンダーの設定

Online Certificate Status Manager の設定時にセキュリティードメイン内の CA が選択される場合は、OCSP サービスを設定する追加の手順は必要ありません。CA の CRL 公開は自動的に設定され、その署名証明書は Online Certificate Status Manager の証明書データベースで自動的に追加および信頼されます。ただし、セキュリティーのないドメイン CA を選択した場合は、Online Certificate Status Manager の設定後に OCSP サービスを手動で設定する必要があります。

注記

OCSP Manager が属するセキュリティードメイン内のすべての CA は、設定時に OCSP Manager によって自動的に信頼されるわけではありません。CA パネルで設定された CA の証明書チェーン内のすべての CA は、OCSP マネージャーによって自動的に信頼されます。セキュリティードメイン内にあるが証明書チェーンにはない他の CA は、手動で信頼させる必要があります。

セキュリティードメイン外の Certificate Manager に Online Certificate Status Manager を設定するには、次を行います。

1. OCSP レスポンダーに公開されるすべての CA に CRL を設定します。
2. OCSP サービスが処理するすべての CA で、公開を有効にし、パブリッシャーを設定し、公開ルールを設定します (9章*証明書および CRL の公開*)。Certificate Manager が LDAP ディレクトリーに公開され、Online Certificated Status Manager がそのディレクトリーから読み込むように設定している場合は、これは必要ありません。
3. 証明書プロファイルは、Certificate Manager が OCSP サービス要求をリッスンする場所を指す Authority Information Access 拡張機能を含むように設定する必要があります (「証明書マネージャーの内部 OCSP サービスの有効化」)。
4. OCSP Responder を設定します。
   - 失効情報ストア (「失効情報ストアの設定: 内部データベース」 および 「失効情報ストアの設定: LDAP ディレクトリー」) を設定します。
   - OCSP レスポンダー (「OCSP レスポンダーへの CA の特定」) へのすべての公開証明書マネージャーを特定します。
   - 必要に応じて、OCSP 署名証明書 (「CA 証明書の信頼設定の変更」) に署名した CA に信頼を設定します。
5. 設定後に両方のサブシステムを再起動します。
6. CA が OCSP レスポンダーに適切に接続されていることを確認します (「証明書マネージャーおよびオンライン証明書ステータスマネージャーの接続の確認」)。

7.6.2. OCSP レスポンダーへの CA の特定

CRL を Online Certificate Status Manager に公開するように CA を設定する前に、Online Certificate Status Manager の内部データベースに CA 署名証明書を保存することにより、CA を Online Certificate Status Manager に識別する必要があります。Certificate Manager は、この証明書に関連するキーペアの CRL を署名します。Online Certificate Status Manager は、保存した証明書に対して署名を検証します。

注記

Online Certificate Status Manager の設定時にセキュリティードメイン内の CA が選択されている場合は、CA を認識するように Online Certificate Status Manager を設定する手順が追加する必要はありません。CA 署名の証明書は自動的に追加され、Online Certificate Status Manager の証明書データベースで信頼されます。ただし、非セキュリティードメイン CA が選択されている場合は、Online Certificate Status Manager を設定した後、CA 署名証明書を証明書データベースに手動で追加する必要があります。

CRL を Online Certificate Status Manager に公開する CA の証明書チェーンをインポートする必要はありません。OCSP サービスに証明書チェーンが必要なのは、CA が CRL を公開するときに SSL/TLS 認証を介して Online Certificate Status Manager に接続する場合のみです。それ以外の場合は、Online Certificate Status Manager に完全な証明書チェーンは必要ありません。

ただし、Online Certificate Status Manager の証明書データベースには、CRL に署名した証明書 (CA 署名証明書または個別の CRL 署名証明書) が必要です。OCSP サービスは、CRL に署名した証明書を、証明書チェーンではなく、データベース内の証明書と比較することにより、CRL を検証します。ルート CA とその下位 CA の 1 つが CRL を Online Certificate Status Manager に公開する場合、Online Certificate Status Manager には両方の CA の CA 署名証明書が必要です。

CA が Online Certificate Status Manager に公開している証明書の署名に使用される CA または CRL 署名証明書をインポートするには、次の手順を実行します。

1. Certificate Manager の base-64 CA 署名証明書は、CA のエンドエンティティーページから取得します。
2. オンライン証明書ステータスマネージャーエージェントページを開きます。URL の形式は **https://hostname:SSLport/ocsp/agent/ocsp** です。
3. 左側のフレームで、**Add Certificate Authority** をクリックします。
4. フォームで、エンコードされた CA 署名証明書を **Base 64 encoded certificate (including the header and footer)** というラベルの付いたテキスト領域内に貼り付けます。
5. 証明書が正常に追加されたことを確認するには、左側のフレームで **List Certificate Authorities** をクリックします。

その結果、新しい CA に関する情報が表示されます。**This Update** フィールド、**Next Update**、および **Requests Served Since Startup** フィールドには、ゼロ (0) の値が表示されます。

7.6.2.1. 証明書マネージャーおよびオンライン証明書ステータスマネージャーの接続の確認

Certificate Manager を再起動すると、Online Certificate Status Manager の SSL/TLS ポートに接続しようとします。Certificate Manager が実際に Online Certificate Status Manager と通信したことを確認するには、**This Update** フィールドおよび **Next Update** フィールドを確認します。これらのフィールドは、CA が Online Certificate Status Manager と最後に通信したときの適切なタイムスタンプで更新する必要があります。クライアントが証明書失効リストのステータスに対して OCSP サービスにクエリーを試行していないため、**Requests Served Since Startup** フィールドにはゼロ (0) の値が表示されるはずです。

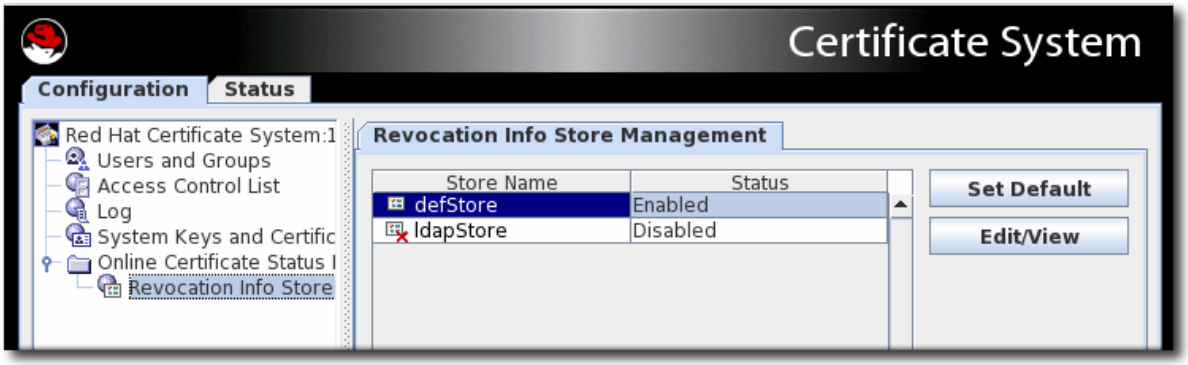
7.6.2.2. 失効情報ストアの設定: 内部データベース

Online Certificate Status Manager は各 Certificate Manager の CRL を内部データベースに保存し、これを CRL ストアとして使用し、証明書の失効ステータスを確認します。Online Certificate Status Manager が CRL を内部データベースに格納するために使用する設定を変更するには、以下を実行します。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。



右側のペインには、Online Certificate Status Manager が使用できる 2 つのリポジトリーが表示されます。デフォルトでは、内部データベースで CRL を使用します。

3. **defStore** を選択して **Edit/View** をクリックします。

4. **defStore** 値を編集します。

- **notFoundAsGood.**問題の証明書が CRL のいずれかに見つからない場合は、GOOD の OCSP 応答を返すように OCSP サービスを設定します。これを選択しないと、応答は UNKNOWN になり、クライアントが発生した場合にはエラーメッセージが表示されます。

- **byName.**OCSP レスポンダーは、応答を行う OCSP レスポンダーの ID を含む基本的な応答タイプのみをサポートします。基本応答タイプの ResponderID フィールドは、***ocsp.store.defStore.byName*** パラメーターの値により決定されます。***byName*** パラメーターが true である、または存在しない場合、OCSP 認証局署名証明書サブジェクト名は OCSP 応答の ResponderID フィールドとして使用されます。***byName*** パラメーターが false の場合、OCSP 認証局署名証明書キーハッシュは OCSP 応答の ResponderID フィールドになります。

- **includeNextUpdate.**次の CRL 更新時間のタイムスタンプが含まれます。

注記

pkiconsole が非推奨になりました。

7.6.2.3. 失効情報ストアの設定: LDAP ディレクトリー

OCSP Manager はデフォルトでは CA CRL を内部データベースに保存しますが、代わりに LDAP ディレクトリーに公開された CRL を使用するよう設定することができます。

重要

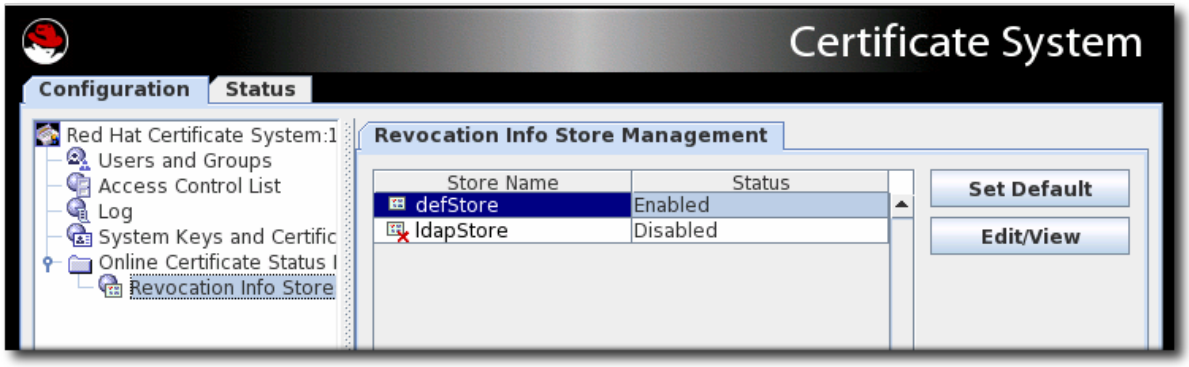
ldapStore メソッドが有効になっていると、OCSP ユーザーインターフェイスは証明書のステータスを確認しません。

LDAP ディレクトリーを使用するように Online Certificate Status Manager を設定するには、以下を実行します。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。

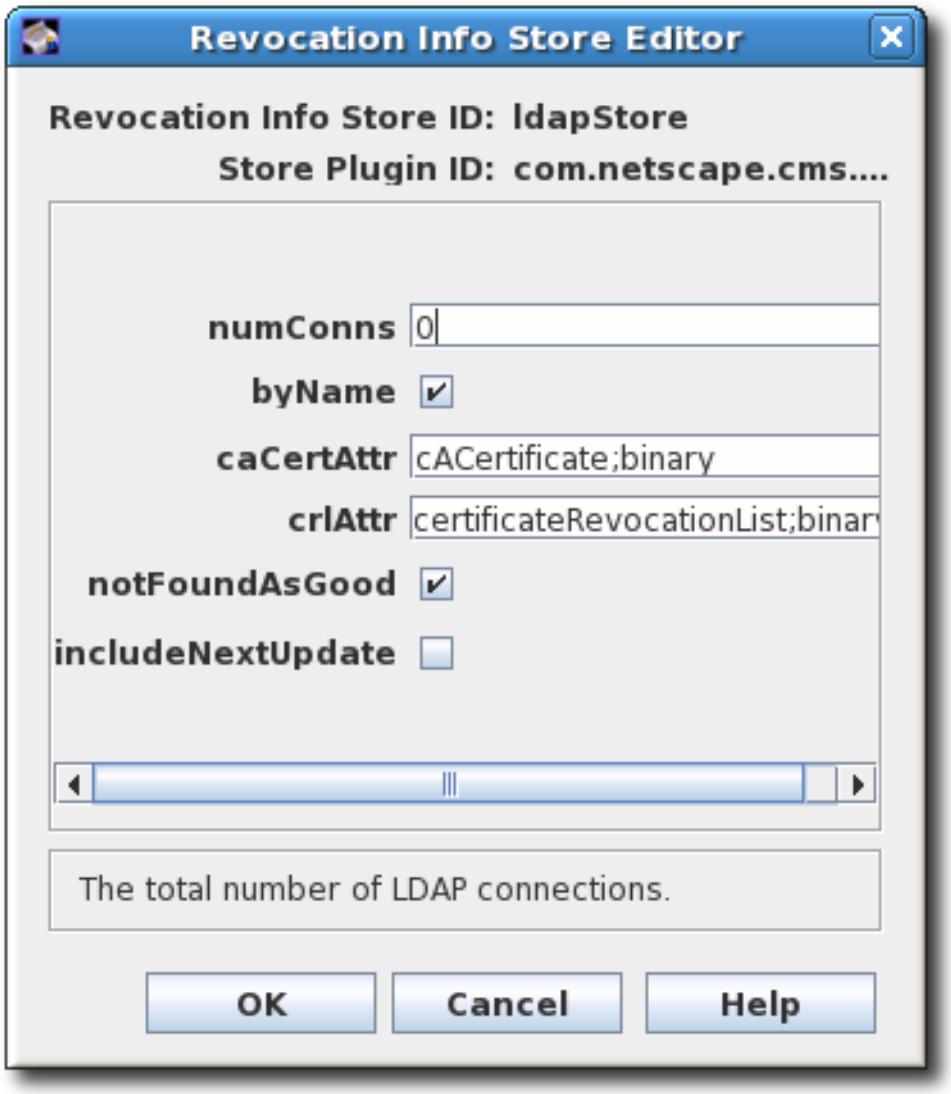


右側のペインには、Online Certificate Status Manager が使用できる 2 つのリポジトリーが表示されます。デフォルトでは、内部データベースで CRL を使用します。

3. LDAP ディレクトリーで CRL を使用するには、**Set Default** をクリックして **ldapStore** オプションを有効にします。

4. **ldapStore** を選択して **Edit/View** をクリックします。

5. **ldapStore** パラメーターを設定します。



- **numConns.**OCSP サービスがチェックする必要のある LDAP ディレクトリーの合計数。デフォルトでは、これは 0 に設定されます。この値を設定すると、対応する**host** フィールド、**port** フィールド、**baseDN** フィールド、および **refreshInSec** フィールドの数が表示されます。

- **host.**LDAP ディレクトリーの完全修飾 DNS ホスト名。

- **port**。LDAP ディレクトリーの SSL/TLS ポート以外のポート。

- **baseDN.**CRL の検索を開始する DN。たとえば、**O=example.com** です。

- **refreshInSec.**接続が更新される頻度。デフォルトは 86400 秒 (毎日) です。

- **caCertAttr.**デフォルト値である **cACertificate;binary** はそのままにしておきます。これは、Certificate Manager がその CA 署名証明書を公開する属性です。

- **crlAttr.**デフォルト値 **certificateRevocationList;binary** はそのままにしておきます。これは、Certificate Manager が CRL を公開する属性です。

- **notFoundAsGood.**問題の証明書が CRL のいずれかに見つからない場合は、GOOD の OCSP 応答を返すように OCSP サービスを設定します。これを選択しないと、応答は UNKNOWN になり、クライアントが発生した場合にはエラーメッセージが表示されます。

- **byName.**OCSP レスポンダーは、応答を行う OCSP レスポンダーの ID を含む基本的な応答タイプのみをサポートします。基本応答タイプの ResponderID フィールド

は、***ocsp.store.defStore.byName*** パラメーターの値により決定されます。***byName*** パラメーターが true である、または存在しない場合、OCSP 認証局署名証明書サブジェクト名は OCSP 応答の ResponderID フィールドとして使用されます。***byName*** パラメーターが false の場合、OCSP 認証局署名証明書キーハッシュは OCSP 応答の ResponderID フィールドになります。

- **includeNextUpdate.**Online Certificate Status Manager には、次の CRL 更新時間のタイムスタンプを含めることができます。

注記

pkiconsole が非推奨になりました。

7.6.2.4. OCSP サービス設定のテスト

以下を実行して、Certificate Manager が OCSP 要求を適切に処理できるかどうかをテストします。

1. ブラウザーまたはクライアントで失効チェックをオンにします。
2. OCSP サービス用に有効になっている CA から証明書を要求します。
3. 要求を承認します。
4. ブラウザーまたはクライアントに証明書をダウンロードします。
5. CA がブラウザーまたはクライアントで信頼されていることを確認します。
6. Certificate Manager の内部 OCSP サービスのステータスを確認します。

   CA エージェントサービスページを開き、**OCSP サービス** のリンクを選択します。
7. 独立した Online Certificate Status Manager サブシステムをテストします。

   Online Certificate Status Manager エージェントサービスページを開き、**List Certificate Authorities** リンクをクリックします。

   このページには、CRL を Online Certificate Status Manager に公開するための設定された Certificate Manager に関する情報が表示されます。このページには、最後に起動した時点の Online Certificate Status Manager のアクティビティーも要約されています。
8. 証明書を取り消します。
9. ブラウザーまたはクライアントで証明書を確認します。サーバーは、証明書が取り消されたことを返す必要があります。
10. Certificate Manager の OCSP サービスステータスを再度チェックして、次のことが発生したことを確認します。

    - ブラウザーは OCSP クエリーを Certificate Manager に送信します。
    - Certificate Manager は OCSP の応答をブラウザーに送信します。
    - ブラウザーはその応答を使用して証明書を検証し、証明書を検証できなかったというステータスを返しました。
11. 独立した OCSP サービスサブシステムを再度チェックし、これらの問題が発生することを確認します。

- 証明書マネージャーは、CRL を Online Certificate Status Manager に公開します。
- ブラウザーは OCSP 応答を Online Certificate Status Manager に送信します。
- Online Certificate Status Manager は OCSP の応答をブラウザーに送ります。
- ブラウザーはその応答を使用して証明書を検証し、証明書を検証できなかったというステータスを返しました。

7.6.3. 問題のあるシリアル番号のレスポンスの設定

OCSP レスポンダーは、証明書が有効かどうかを判断する前に、証明書の失効ステータスと有効期限を確認します。デフォルトでは、OCSP は証明書の他の情報を検証しません。

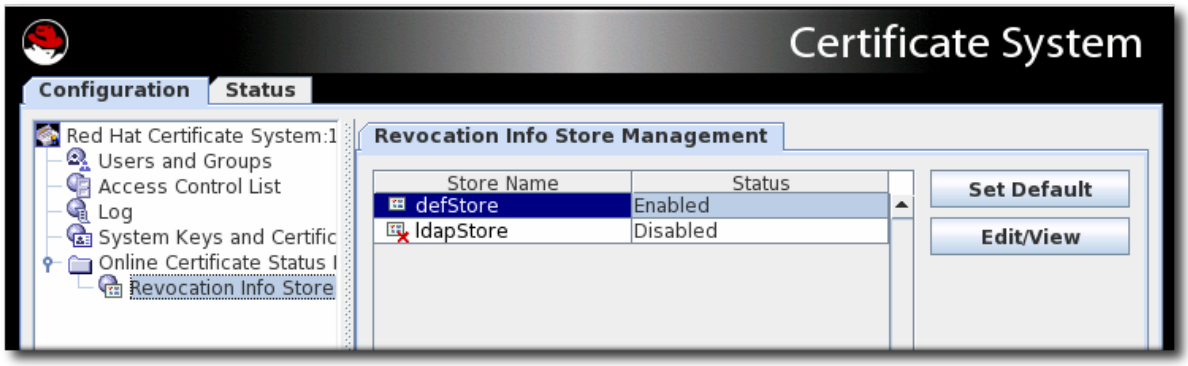
notFoundAsGood パラメーターは、OCSP が無効なシリアル番号で証明書を処理する方法を設定します。このパラメーターはデフォルトで有効になっています。つまり、証明書が不正なシリアル番号で存在する場合は、証明書が有効であれば、OCSP が証明書の **GOOD** のステータスを返します。

OCSP に、不正なシリアル番号と失効ステータスに基づいて証明書をチェックおよび拒否させるには、***notFoundAsGood*** 設定を変更します。この場合、OCSP は、間違ったシリアル番号を持つ証明書とともに **UNKNOWN** ステータスを返します。クライアントはエラーとして解釈し、それに応じて応答できます。

1. オンライン証明書ステータスマネージャーコンソールを開きます。

```
pkiconsole https://server.example.com:8443/ocsp
```

2. **Configuration** タブで **Online Certificate Status Manager** を選択し、**Revocation Info Stores** を選択します。



3. **defStore** を選択して **Edit/View** をクリックします。
4. ***notFoundAsGood*** 値を編集します。このチェックボックスを選択すると、証明書のシリアル番号が不正な場合でも OCSP が **GOOD** の値を返します。チェックボックスの選択を解除すると、OCSP は、**UNKNOWN** の値を送信します。クライアントはこれをエラーとして解釈できます。

5. OCSP Manager を再起動します。

```
]# pki-server restart instance-name
```

注記

pkiconsole が非推奨になりました。

7.6.4. 証明書マネージャーの内部 OCSP サービスの有効化

Certificate Manager には、OCSP 準拠のクライアントでビルトインの OCSP サービスがあり、Certificate Manager に、証明書の失効ステータスを直接問い合わせることができます。Certificate Manager がインストールされると、OCSP 署名証明書が発行され、OCSP サービスがデフォルトで有効になります。この OCSP 署名証明書は、OCSP サービスリクエストへのすべての応答に署名するために使用されます。内部 OCSP サービスは、Certificate Manager の内部データベースに格納されている証明書のステータスをチェックするため、このサービスを使用するように公開を設定する必要はありません。

クライアントは、Certificate Manager の SSL/TLS エンドエンティティーポートを介して OCSP サービスをクエリーできます。証明書失効ステータスをクエリーすると、Certificate Manager は証明書の内部データベースを検索し、そのステータスを確認してクライアントに応答します。Certificate Manager は発行されたすべての証明書のリアルタイムステータスであるため、失効確認の方法は最も正確です。

インストール時に、すべての CA ビルトイン OCSP サービスが有効になっている。ただし、このサービスを使用するには、CA が Authority Information Access 拡張で証明書を発行する必要があります。

1. CA のエンドエンティティーに移動します。以下に例を示します。

```
https://server.example.com:8443/ca/ee/ca
```

2. CA 署名証明書を探します。

3. 証明書で Authority Info Access 拡張を探し、**https://server.example.com:8443/ca/ocsp** などの **Location URIName** 値をメモします。

4. 登録プロファイルを更新して、Authority Information Access 拡張を有効にし、**Location** パラメーターを Certificate Manager の URI に設定します。証明書プロファイルの編集に関する詳細は、「証明書プロファイルの設定」 を参照してください。

5. CA インスタンスを再起動します。

```
]# pki-server restart instance-name
```

注記

Certificate Manager の内部 OCSP サービスを無効にするには、CA の **CS.cfg** ファイルを編集し、**ca.ocsp** パラメーターの値を **false** に変更します。

```
ca.ocsp=false
```

7.6.5. OCSPClient プログラムを使用した OCSP リクエストの送信

OCSPClient プログラムは、OCSP リクエストの実行に使用できます。以下に例を示します。

```
]# OCSPClient -h server.example.com -p 8080 -d /etc/pki/pki-tomcat/alias -c "caSigningCert cert-pki-ca" --serial 2
CertID.serialNumber=2
CertStatus=Good
```

OCSPClient コマンドは、以下のコマンドラインオプションと共に使用できます。

表7.1 利用可能な OCSPClient オプション

オプション	説明
-d *database*	セキュリティーデータベースの場所 (デフォルト: 現行ディレクトリー)
-h *hostname*	OCSP サーバーのホスト名 (デフォルト: example.com)
-p *port*	OCSP サーバーのポート番号 (デフォルト: 8080)
-t *path*	OCSP サービスパス (デフォルト: /ocsp/ee/ocsp)

オプション	説明
-c *nickname*	CA 証明書のニックネーム (デフォルト: CA 署名証明書)
-n *times*	送信番号 (デフォルトは 1)
--serial *serial_number*	チェックする証明書のシリアル番号
--input *input_file*	DER でエンコードされた OCSP 要求が含まれる入力ファイル
--output *output_file*	DER でエンコードされた OCSP 応答を保存する出力ファイル
-v, --verbose	詳細モードで実行
--help	ヘルプメッセージを表示

7.6.6. GET メソッドを使用した OCSP リクエストの送信

RFC 6960 で説明されているように、255 バイト未満の OCSP 要求は、GET メソッドを使用して Online Certificate Status Manager に送信できます。GET 経由で OCSP 要求を送信するには、以下のコマンドを実行します。

1. クエリーされるステータスで、証明書の OCSP 要求を生成します。以下に例を示します。

```
]# openssl ocsp -CAfile ca.pem -issuer issuer.pem -serial serial_number -reqout - | base64

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. Web ブラウザーのアドレスバーに URL を貼り付けて、ステータス情報を返します。ブラウザーが OCSP 要求を処理できるようにする必要があります。

```
https://server.example.com:8443/ocsp/ee/ocsp/MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4
cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy35o7xW5BMzM8FTvyTwCAQE=
```

3. OCSP Manager は、ブラウザーが解釈できる証明書のステータスを返します。設定可能なステータスは GOOD、REVOKED、および UNKNOWN です。

あるいは、**curl** などのツールを使用して、要求と **openssl** を使用して応答を解析して、コマンドラインから OCSP を実行します。以下に例を示します。

1. クエリーされるステータスで、証明書の OCSP 要求を生成します。以下に例を示します。

```
]# openssl ocsp -CAfile ca.pem -issuer issuer.pem -serial serial_number -reqout - | base64

MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy
35o7xW5BMzM8FTvyTwCAQE=
```

2. **curl** を使用して、OCSP 要求を送信するために OCSP Manager に接続します。

```
curl
https://server.example.com:8443/ocsp/ee/ocsp/MEIwQDA+MDwwOjAJBgUrDgMCGgUABBT4
cyABkyiCIhU4JpmIBewdDnn8ZgQUbyBZ44kgy35o7xW5BMzM8FTvyTwCAQE= >
ocspresp.der
```

3. **openssl** を使用して応答を解析します。

```
openssl ocsp -respin ocspresp.der -resp_text
```

Authority Information Access 拡張機能を備えた 7.1 CA によって発行された証明書を GET メソッドを使用して OCSP に送信するには、「Certificate System 7.1 以前で発行された証明書のリダイレクトの設定」で説明するように、要求を適切な URL に転送するためのリダイレクトを作成する必要があります。

7.6.7. Certificate System 7.1 以前で発行された証明書のリダイレクトの設定

ファイルルート **/ocsp/ee/ocsp/** を含む URL で指定された OCSP ユーザーページの場所は、Certificate System 10 または Certificate System 8.1 と、Certificate System 7.1 とで異なります (Certificate System 7.1 では **/ocsp/**)。Authority Information Access 拡張機能を備えた 7.1 以前の CA によって発行された証明書を OCSP に送信するには、リダイレクトを作成して、要求を適切な URL に転送します。

注記

リダイレクトの設定は、Authority Information Access 拡張機能を備えた 7.1 以前の CA によって発行された証明書を管理するためにのみ必要です。証明書が新しいバージョンの Certificate Manager によって発行されている場合、または Authority Information Access 拡張機能が含まれていない場合、この設定は必要ありません。

1. OCSP レスポンダーを停止します。

```
]# pki-server stop instance-name
```

2. OCSP のエンドユーザー Web アプリケーションディレクトリーに移動します。以下に例を示します。

```
]# cd /var/lib/pki-ocsp/webapps/ocsp
```

3. OCSP の Web アプリケーションディレクトリーの **ROOT/WEB-INF/** ディレクトリーにある **ROOT** ディレクトリーに移動します。以下に例を示します。

```
]# cd /var/lib/pki-ocsp/webapps/ocsp/ROOT/WEB-INF/
```

4. OCSP の Web アプリケーションディレクトリーの **ROOT** ディレクトリーに **lib/** ディレクトリーを作成して開きます。

```
]# mkdir lib
]# cd lib/
```

5. **/usr/share/java/pki/cms.jar** JAR ファイルへリンクするシンボリックリンクを作成します。以下に例を示します。

```
-
```

```
]# ln -s /usr/share/java/pki/cms.jar cms.jar
```

6. メインの Web アプリケーションディレクトリーに移動します。以下に例を示します。

```
]# cd /var/lib/pki-ocsp/webapps/ocsp/
```

7. 現行インスタンス (**ocsp**) ディレクトリーの名前を変更します。以下に例を示します。

```
]# mv /var/lib/pki-ocsp/webapps/ocsp/ocsp /var/lib/pki-ocsp/webapps/ocsp/ocsp2
```

8. 元の **ocsp**/ ディレクトリーの **WEB-INF**/ ディレクトリーに移動します。以下に例を示します。

```
]# cd  /var/lib/pki-ocsp/webapps/ocsp/ocsp/WEB-INF
```

9. 元の **ocsp**/**WEB-INF**/ ディレクトリーで、**web.xml** ファイルを編集し、**eeocspAddCRL** と **csadmin-wizard** サーブレットの間に行マッピングを追加します。

```
 <servlet-mapping>
   <servlet-name>  ocspOCSP  </servlet-name>
   <url-pattern>   /ee/ocsp/*  </url-pattern>
 </servlet-mapping>
```

10. **ROOT** ディレクトリーに **web.xml** ファイルを作成し、インストールします。以下に例を示します。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app>

 <display-name>Welcome to Tomcat</display-name>
 <description>
   Welcome to Tomcat
 </description>

 <servlet>
  <servlet-name>ocspProxy</servlet-name>
  <servlet-class>com.netscape.cms.servlet.base.ProxyServlet</servlet-class>
  <init-param>
   <param-name>destContext</param-name>
   <param-value>/ocsp2</param-value>
  </init-param>
  <init-param>
   <param-name>destServlet</param-name>
   <param-value>/ee/ocsp</param-value>
  </init-param>
 </servlet>

 <servlet>
  <servlet-name>ocspOther</servlet-name>
  <servlet-class>com.netscape.cms.servlet.base.ProxyServlet</servlet-class>
  <init-param>
   <param-name>destContext</param-name>
   <param-value>/ocsp2</param-value>
  </init-param>
```

```
    <init-param>
      <param-name>srcContext</param-name>
      <param-value>/ocsp</param-value>
    </init-param>
    <init-param>
      <param-name>destServlet</param-name>
      <param-value></param-value>
    </init-param>
    <init-param>
      <param-name>matchURIStrings</param-name>

<param-value>/ocsp/registry,/ocsp/acl,/ocsp/jobsScheduler,/ocsp/ug,/ocsp/server,/ocsp/log,
          /ocsp/auths,/ocsp/start,/ocsp/ocsp,/ocsp/services,/ocsp/agent,/ocsp/ee,
          /ocsp/admin</param-value>
    </init-param>
    <init-param>
      <param-name>destServletOnNoMatch</param-name>
      <param-value>/ee/ocsp</param-value>
    </init-param>
    <init-param>
      <param-name>appendPathInfoOnNoMatch</param-name>
      <param-value>/ocsp</param-value>
    </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>ocspProxy</servlet-name>
    <url-pattern>/ocsp</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>ocspOther</servlet-name>
    <url-pattern>/ocsp/*</url-pattern>
  </servlet-mapping>

</web-app>
```

11. **/var/lib/pki-ocsp/conf/context.xml** ファイルを編集して、以下の行を追加します。

```
<Context>
 to
<Context crossContext="true" >
```

12. **/var/lib/pki-ocsp/webapps/ocsp/ocsp2/services.template** ファイルを編集し、以下の行を変更します。

```
result.recordSet[i].uri);
 to
result.recordSet[i].uri + "/");
```

13. OCSP インスタンスを起動します。

```
]# pki-server start instance-name
```

第8章 PKI ACME RESPONDER の管理

本章では、**PKI ACME Responder** を管理する方法を説明します。

PKI ACME Responder の設定方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『PKI ACME Responder の設定』の章を参照してください。

8.1. ACME サービスの有効化/無効化

Administrators グループに属するユーザーは、ACME レスポンダーでサービスを有効または無効にすることができます。ユーザーは、Basic 認証またはクライアント証明書認証のいずれかで認証できます。

- Basic 認証で ACME サービスを有効または無効にするには、ユーザー名とパスワードを指定します。

```
$ pki -u <username> -p <password> acme-<enable/disable>
```

- クライアント証明書認証で ACME サービスを有効または無効にするには、証明書のニックネームと NSS データベースのパスワードを指定します。

```
$ pki -n <nickname> -c <password> acme-<enable/disable>
```

8.2. PKI ACME RESPONDER のステータスの確認

- ACME レスポンダーのステータスを確認するには、以下のコマンドを実行します。

```
$ pki acme-info
Status: Available
Terms of Service: https://www.example.com/acme/tos.pdf
Website: https://www.example.com
CAA Identities: example.com
External Account Required:false
```

サービスが無効になっていると、コマンドにより以下の結果が表示されます。

```
$ pki acme-info
Status: Unavailable
```

注記

実際の出力は、**metadata.conf** 設定ファイルで設定した内容により異なります。

パート III. CA サービスを管理するための追加設定

第9章 証明書および CRL の公開

Red Hat Certificate System には、Certificate Manager 用のカスタマイズ可能な公開フレームワークが含まれており、証明書機関は、証明書、証明書失効リスト (CRL)、およびその他の証明書関連オブジェクトを、サポートされているリポジトリー (LDAP 準拠のディレクトリー、フラットファイル、およびオンライン検証機関) に有効にします。本章では、証明書および CRL をファイル、ディレクトリー、および Online Certificate Status Manager に公開するように Certificate Manager を設定する方法を説明します。

パブリッシュを設定する一般的なプロセスは次のとおりです。

1. ファイル、LDAP ディレクトリー、または OCSP レスポンダーへの公開を設定します。

   使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

2. ルールを設定して、どの証明書がその場所に公開されるかを決定します。証明書または CRL が一致するすべてのルールがアクティブ化されるため、ファイルベースのルールとディレクトリーベースのルールを一致させることにより、同じ証明書をファイルと LDAP ディレクトリーに公開できます。

   ルールは、各オブジェクトタイプ (CA 証明書、CRL、ユーザー証明書、およびクロスペアの証明書) に設定できます。使用されていないルールをすべて無効にします。

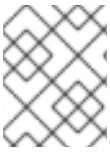
3. CRL を設定します。CRL は公開前に設定する必要があります。*7章証明書の取り消しおよび CRL 発行* を参照してください。

4. パブリッシャー、マッパー、およびルールの設定後に公開を有効にします。公開が有効になると、サーバーはすぐに公開を開始します。パブリッシャー、マッパー、およびルールが完全に設定されていない場合は、パブリッシュが正しく機能しない可能性があります。

9.1. 公開の概要

証明書システムは、ファイルまたは LDAP ディレクトリーに証明書を公開したり、CRL をファイル、LDAP ディレクトリー、OCSP レスポンダーに公開したりできます。

柔軟性を高めるために、特定のタイプの証明書または CRL を単一の形式または 3 つすべての形式で公開できます。たとえば、CA 証明書はディレクトリーにのみ公開され、ファイルには公開されず、ユーザー証明書はファイルとディレクトリーの両方に公開できます。



注記

OCSP レスポンダーは CRL に関する情報のみを提供します。証明書は OCSP レスポンダーに公開されません。

証明書ファイルと CRL ファイルに異なる公開場所を設定でき、さまざまな種類の証明書ファイルや異なるタイプの CRL ファイルとの間で異なる公開場所を設定することができます。

同様に、異なるタイプの証明書や異なるタイプの CRL をディレクトリー内の異なる場所に公開できます。たとえば、所属企業の West Coast 部門からの証明書は、ディレクトリーの 1 つのブランチで公開することができますが、East Coast 部門のユーザーの証明書をディレクトリー内の他のブランチに公開することができます。

公開が有効になっている場合、証明書または CRL が発行、更新、または取り消されるたびに、公開システムが呼び出されます。証明書または CRL はルールによって評価され、ルールのタイプおよび述語と一致するかどうかを確認します。タイプは、オブジェクトが CRL、CA 証明書、またはその他の証明書であるかどうかを指定します。述語は、評価されるオブジェクトのタイプに対してさらに基準を設定します。たとえば、ユーザー証明書を指定するか、West Coast ユーザー証明書を指定できます。述語を使用するには、公開ルールの述語フィールドに値を入力する必要があります。また、対応する値 (形式は多少異なります) を証明書または証明書要求に含める必要があります。証明書または証明書要求の値は、証明書のタイプなどの証明書の情報から取得することも、要求フォームに配置された非表示の値から取得することもできます。述語が設定されていない場合は、そのタイプのすべての証明書が一致することが考慮されます。たとえば、**CRL** がタイプとして設定されている場合、すべての CRL がルールに一致します。

マッチするすべてのルールは、そのルールで指定された方法および場所に従って証明書または CRL を公開します。指定された証明書または CRL は、ルール、複数のルール、またはすべてのルールに一致しません。公開システムは、発行されたすべての証明書と CRL をすべてのルールと照合しようとします。

ルールがマッチすると、そのルールに関連するパブリッシャーに指定されたメソッドおよび場所に従って、証明書または CRL が公開されます。たとえば、ルールがユーザーに発行されたすべての証明書と一致して、ルールにその場所 **/etc/CS/certificates** のファイルに公開する発行者がある場合、証明書はファイルとしてその場所に公開されます。別のルールが、ユーザーに発行されたすべての証明書に一致し、そのルールに LDAP 属性 **userCertificate;binary** 属性に公開するパブリッシャーがある場合、証明書は、ユーザーのエントリーのこの属性で LDAP 公開が有効になったときに指定されたディレクトリーに発行されます。

ファイルに公開するように指定するルールの場合、証明書または CRL が古くなったディレクトリーに新しいファイルが作成されます。

LDAP ディレクトリーに公開するように指定するルールの場合、指定された属性に、証明書または CRL がディレクトリーに指定されたエントリーに公開されます。CA は、公開された証明書または CRL 属性の値を後続の証明書または CRL で上書きします。簡単に言うと、すでに公開されている既存の証明書または CRL は、次の証明書または CRL に置き換えられます。

Online Certificate Status Manager への公開を指定するルールの場合、CRL はこのマネージャーに公開されます。証明書は Online Certificate Status Manager に公開されません。

LDAP 公開の場合は、ユーザーのエントリーの場所を決定する必要があります。マッパーは、公開するエントリーを決定するために使用されます。マッパーには、エントリーの正確な DN、証明書から取得した DN を作成するための情報を関連付けた変数、あるいはディレクトリーを検索してエントリー内の一意の属性や属性のセットを検索し、エントリーの正しい DN を確認するための十分な情報を含めることができます。

証明書が取り消されると、サーバーは公開ルールを使用して、LDAP ディレクトリーまたはファイルシステムから対応する証明書を見つけて削除します。

証明書の有効期限が切れると、サーバーは設定されたディレクトリーからその証明書を削除できます。サーバーはこれを自動的に実行しません。適切なジョブを実行するようにサーバーを設定する必要があります。詳細は、13章 *自動ジョブの設定* を参照してください。

公開の設定には、パブリッシャー、マッパー、およびルールを設定する必要があります。

9.1.1. パブリッシャー

パブリッシャー は、証明書と CRL が公開される場所を指定します。ファイルに公開する場合、パブリッシャーはファイルシステムの公開ディレクトリーを指定します。LDAP ディレクトリーに公開する場合、発行者は証明書または CRL を格納するディレクトリーの属性を指定します。マッパーは、エン

トリーの DN を決定するために使用されます。DN ごとに、その DN を導出するための異なる式が設定されます。LDAP 公開が有効であるときに LDAP ディレクトリーの場所が指定されます。OCSP レスポンダーに CRL を公開する場合、パブリッシャーは Online Certificate Status Manager のホスト名と URI を指定します。

9.1.2. マッパー

マッパー は LDAP 公開でのみ使用されます。マッパーは、証明書または証明書要求からの情報に基づいて、エントリーの DN を構築します。サーバーには、証明書のサブジェクト名および証明書要求の情報があり、この情報を使用してそのエントリーの DN を作成する方法を把握する必要があります。マッパーは、利用可能な情報を DN、またはディレクトリー内で検索してエントリーの DN を取得できる一意の情報に変換するための式を提供します。

9.1.3. ルール

ファイル、LDAP、および OCSP 公開の **ルール** は、証明書または CRL を公開するかどうかとその方法をサーバーに指示します。ルールは、最初に、ルールのタイプと述語を設定することにより、公開されるもの、特定の特性に一致する証明書または CRL を定義します。次に、ルールは、パブリッシャーに関連付けられ、LDAP 公開の場合はマッパーに関連付けられることにより、公開方法と場所を指定します。

ルールは、PKI デプロイメントに必要なだけ単純または複雑にすることができ、さまざまなシナリオに対応するのに十分な柔軟性があります。

9.1.4. ファイルへの公開

サーバーは、証明書と CRL をフラットファイルに公開できます。フラットファイルは、リレーショナルデータベースなどの任意のリポジトリーにインポートできます。サーバーが証明書および CRL をファイルに公開するように設定されている場合、公開ファイルは DER でエンコードされたバイナリーブロブ、base-64 でエンコードされたテキストブロブ、またはその両方になります。

- サーバーの問題の各証明書について、DER でエンコードされた形式または base-64 でエンコードされた形式で、証明書が含まれるファイルを作成します。各ファイルには **cert-serial_number.der** または **cert-serial_number.b64** という名前が付けられます。**serial_number** は、ファイルに含まれる証明書のシリアル番号です。たとえば、シリアル番号が **1234** の DER でエンコードされた証明書のファイル名は、 **cert-1234.der** です。

- サーバーが CRL を生成するたびに、DER でエンコードされた形式または base-64 でエンコードされた形式で新しい CRL を含むファイルが作成されます。各ファイルの名前は、形式に応じて、**issuing_point_name-this_update.der** または **issuing_point_name-this_update.b64** のいずれかになります。**issuing_point_name** は CRL を公開する CRL 発行ポイントを識別します。**this_update** は、ファイルに含まれる CRL のタイム依存更新値から取得する値を指定します。たとえば、値が **This Update: Friday January 28 15:36:00 PST 2020** である DER でエンコードされた CRL のファイル名は **MasterCRL-20200128-153600.der** です。

9.1.5. OCSP 公開

Certificate System OCSP サービスには、Certificate Manager の内部サービスと Online Certificate Status Manager の 2 つの形式があります。内部サービスは、Certificate Manager の内部データベースを確認して、証明書のステータスを報告します。内部サービスは公開用に設定されていません。内部データベースに格納されている証明書を使用して、証明書のステータスを判別します。Online Certificate Status Manager は、Certificate Manager によって送信される CRL を確認します。パブリッシャーは、CRL が送信される場所ごとに設定され、送信される各タイプの CRL に対して 1 つのルールが設定されます。

両方の OCSP サービスの詳細は、「OCSP (Online Certificate Status Protocol) レスポンダーの使用」を参照してください。

9.1.6. LDAP 公開

LDAP の公開 では、サーバーは LDAP または LDAPS を使用して証明書、CRL、およびその他の証明書関連のオブジェクトをディレクトリーに公開します。公開するディレクトリーのブランチは、*公開ディレクトリー* と呼ばれます。

- サーバーが発行する証明書ごとに、ユーザーのエントリーの指定された属性に DER エンコード形式の証明書を含むブロブが作成されます。証明書は DER でエンコードされたバイナリーブロブとして公開されます。
- サーバーが CRL を生成するたびに、CA のエントリーの指定された属性で、DER でエンコードされた形式で新しい CRL を含むブロブを作成します。

LDAP プロトコルまたは SSL (LDAPS) プロトコルを使用して、証明書および CRL を LDAP 準拠のディレクトリーに公開し、アプリケーションは HTTP 経由で証明書および CRL を取得できます。HTTP で証明書および CRL の取得をサポートすると、一部のブラウザーはサーバーから通常の更新を受け取るディレクトリーから最新の CRL を自動的にインポートできます。ブラウザーは CRL を使用してすべての証明書を自動的にチェックし、証明書が取り消されていないことを確認できます。

LDAP 公開が機能するには、ユーザーエントリーが LDAP ディレクトリーに存在する必要があります。

サーバーと公開ディレクトリーが何らかの理由で同期しなくなった場合は、特権ユーザー (管理者とエージェント) も手動で公開プロセスを開始できます。手順は、「ディレクトリーでの CRL の手動による更新」を参照してください。

9.2. ファイルへの公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

ファイルへの公開は、CRL または証明書を特定のホスト上のテキストファイルに公開するだけです。

パブリッシャーは、発行場所ごとに作成および設定する必要があります。パブリッシャーは、ファイルに公開するために自動的に作成されません。すべてのファイルを単一の場所に公開するには、パブリッシャーを 1 つ作成します。異なる場所に公開するには、各場所にパブリッシャーを作成します。場所には、ユーザー証明書などのオブジェクトタイプ、または West Coast ユーザー証明書などのオブジェクトタイプのサブセットを含めることができます。

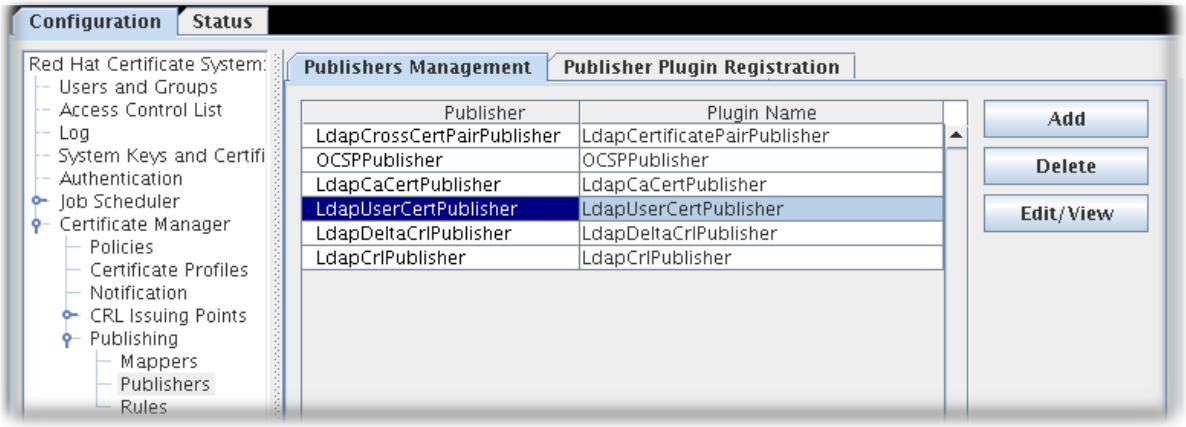
ファイルに公開するためのパブリッシャーを作成するには、以下の手順を実施します。

1. Certificate Manager コンソールにログインします。

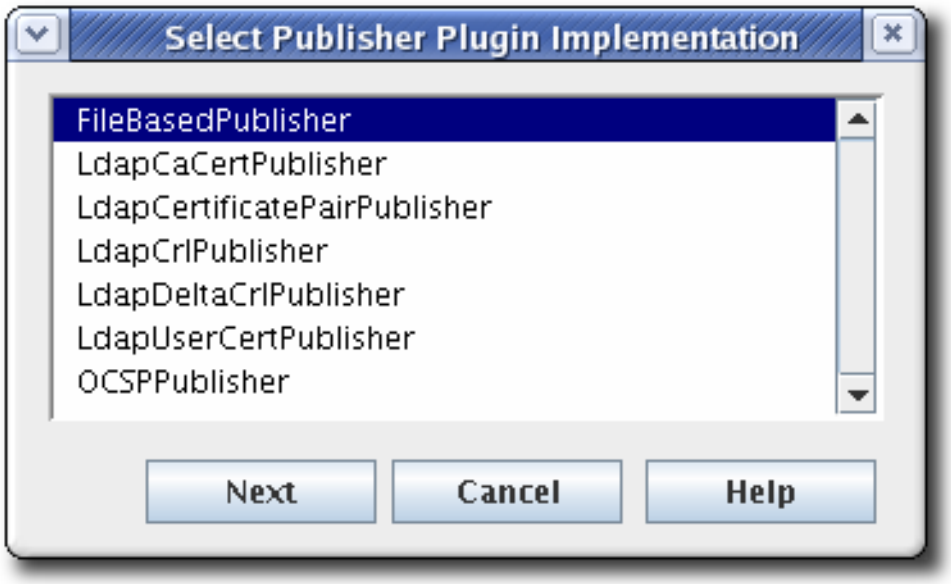
   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Publishers** を選択します。

   設定されたパブリッシャーインスタンスを一覧表示する **Publishers Management** タブが右側で開きます。

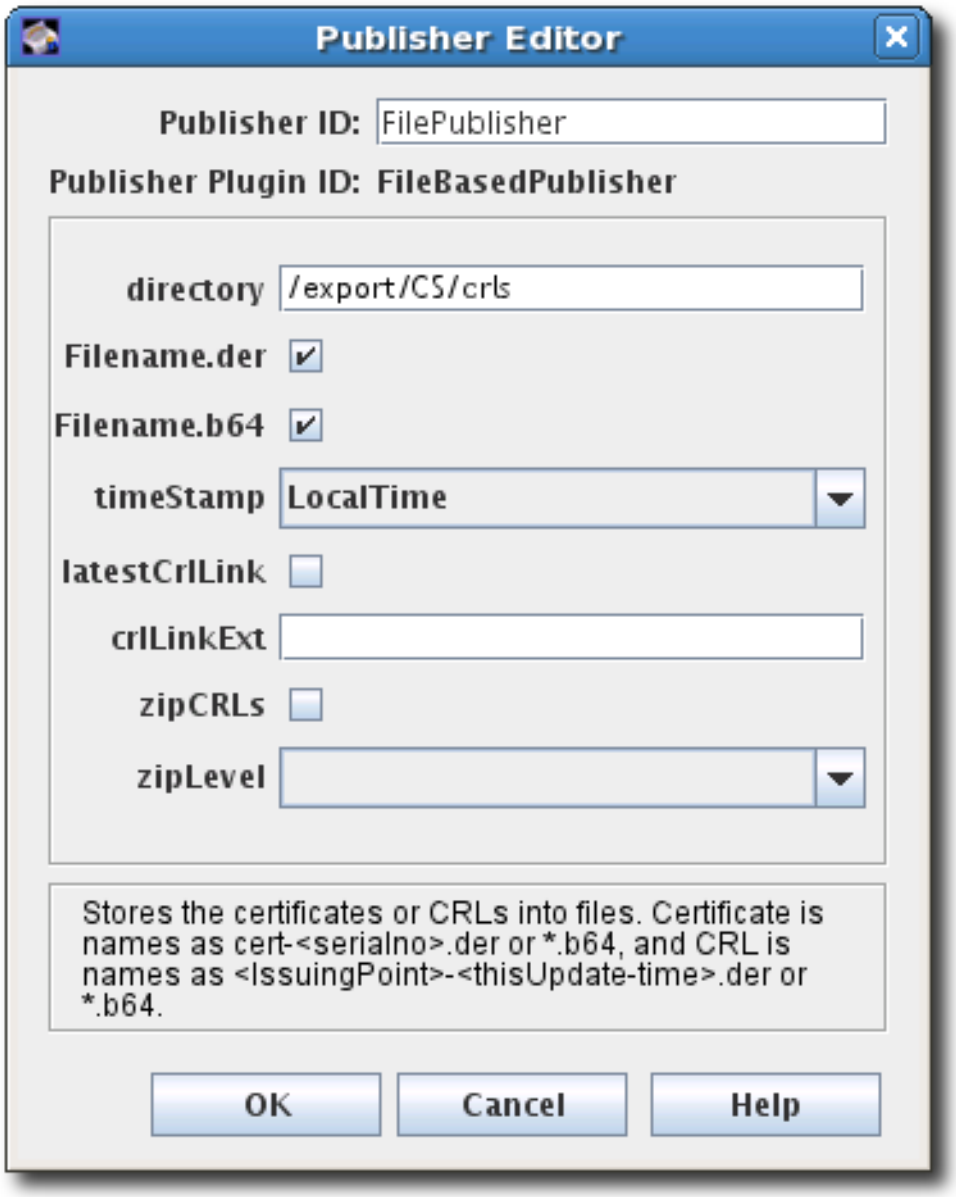


3. **Add** をクリックして、**Select Publisher Plug-in Implementation** ウィンドウを開きます。これには、登録済みのパブリッシャーモジュールを一覧表します。



4. **FileBasedPublisher** モジュールを選択して、エディターウィンドウを開きます。

   これは、Certificate Manager が証明書をファイルに公開し、CRL をファイルに公開できるようにするモジュールです。



5. 証明書を公開するための情報を設定します。

    - **PublishCertsToFile** などの空白のない英数字の発行側の ID
    - Certificate Manager がファイルを公開する必要のあるディレクトリーへのパス。パスは絶対パスを指定でき、Certificate System インスタンスのディレクトリーに相対することもできます。たとえば、**/export/CS/certificates** です。
    - DER でエンコードされたファイル、base-64 でエンコードされたファイル、またはその両方のチェックボックスを選択して公開するファイルタイプ。
    - CRL の場合は、タイムスタンプの形式です。公開される証明書にはファイル名にシリアル番号が含まれ、CRL はタイムスタンプを使用します。
    - CRL の場合、最新の CRL に移動するためにファイルにリンクを生成するかどうか。有効にすると、リンクは、拡張機能で使用する CRL 発行ポイントの名前が **crlLinkExt** フィールドに指定されると想定します。
    - CRL の場合、CRL を圧縮 (zip) するかどうか、および使用する圧縮レベル。

パブリッシャーを設定した後、「ルールの作成」 の説明に従って、発行された証明書と CR L のルールを設定します。

注記

pkiconsole が非推奨になりました。

9.3. OCSP への公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

OCSP Manager への公開は、クライアント検証のために CRL を特定の場所に公開することです。

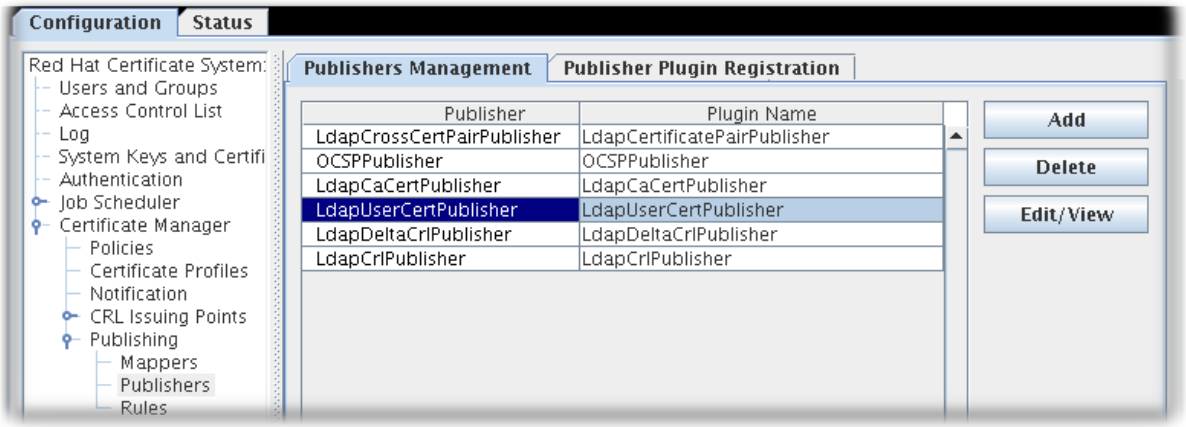
パブリッシャーは、公開場所ごとに作成および設定する必要があります。パブリッシャーは、OCSP レスポンダーに公開するために自動的に作成されません。単一のパブリッシャーを作成して、すべての場所を 1 つの場所に公開するか、CRL を公開するすべての場所のパブリッシャーを作成します。各場所には、さまざまな種類の CRL を含めることができます。

9.3.1. クライアント認証を使用した OCSP への公開の有効化

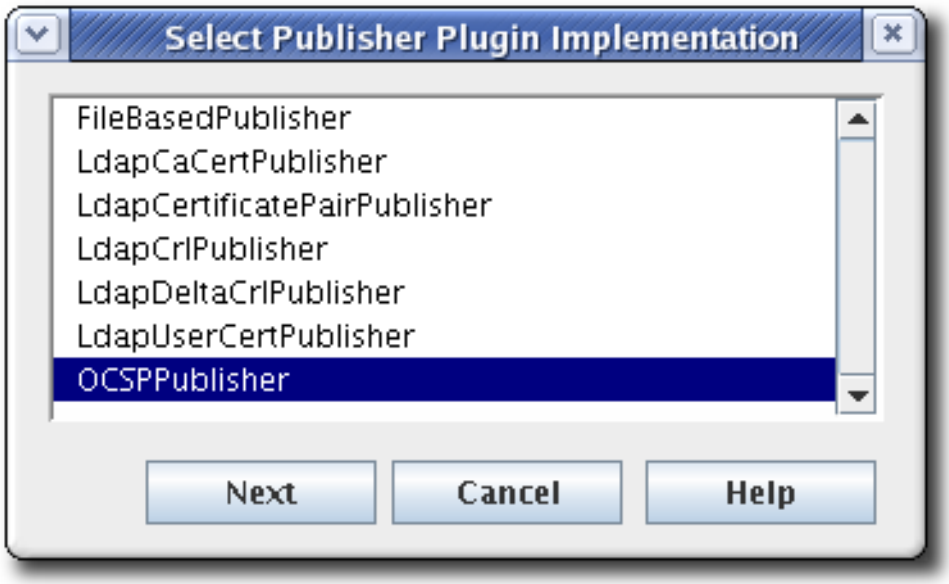
1. Certificate Manager コンソールにログインします。

```
pkiconsole https://server.example.com:8443/ca
```

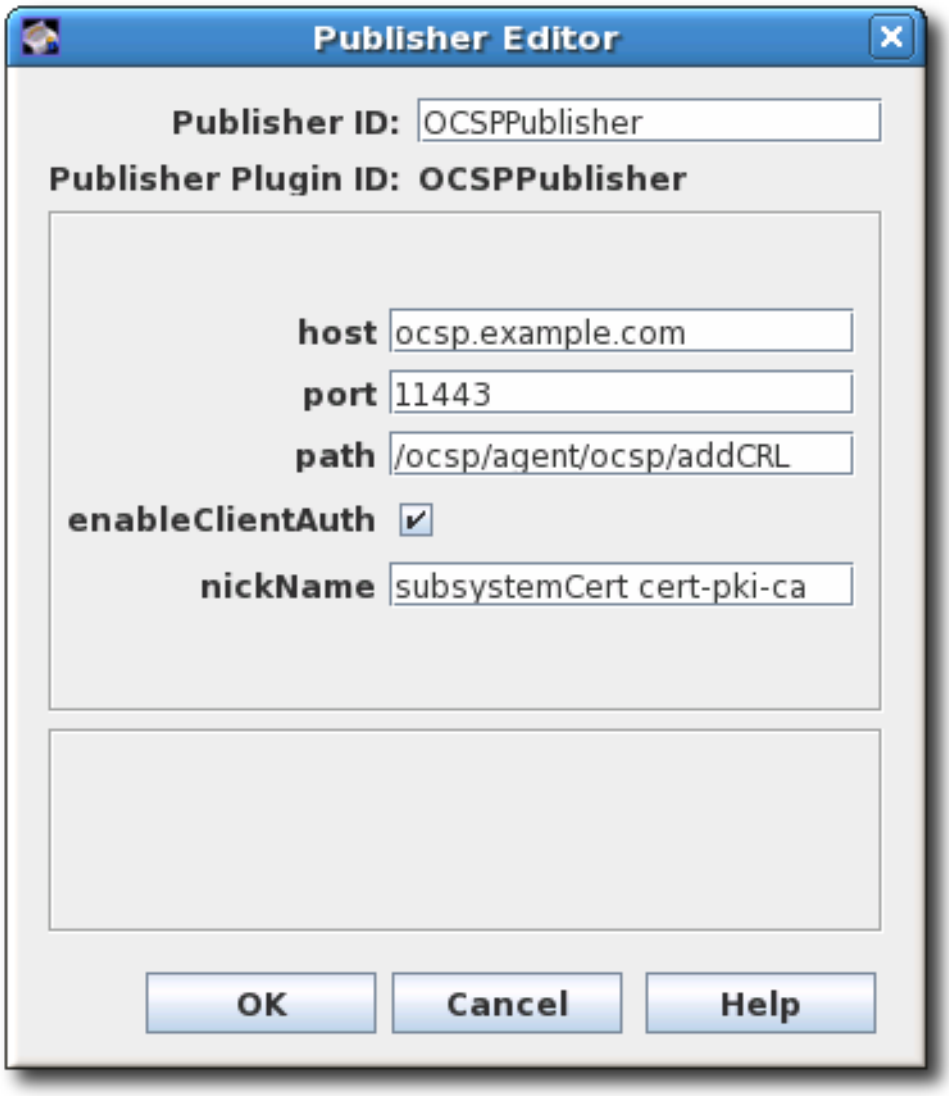
2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Publishers** を選択します。



3. **Add** をクリックして、**Select Publisher Plug-in Implementation** ウィンドウを開きます。これには、登録済みのパブリッシャーモジュールを一覧表します。



4. **OCSPPublisher** モジュールを選択し、エディターウィンドウを開きます。これは、Certificate Manager が CRL を Online Certificate Status Manager に公開できるようにするパブリッシャーモジュールです。



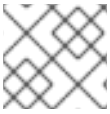
- パブリッシャー ID は、**PublishCertsToOCSP** のように、スペースのない英数字の文字列である必要があります。

- **host** は、**ocspResponder.example.com** または IPv4 または IPv6 アドレスなどの完全修飾ドメイン名を使用できます。
- デフォルトのパスは、**/ocsp/agent/ocsp/addCRL** のように CRL を送信するディレクトリーです。
- クライアント認証が使用されている (**enableClientAuth** が選択されている) 場合は、**nickname** フィールドに認証に使用する証明書のニックネームを指定します。この証明書は OCSP セキュリティーデータベースに存在している必要があります。これは通常 CA サブシステム証明書です。

5. OCSP Manager で CA のユーザーエントリーを作成します。ユーザーは、新しい CRL を送信するときに OCSP への認証に使用されます。必要なものは 2 つあります。
   - **CA-hostname-EEport** などの CA サーバーの後に OCSP ユーザーエントリーに名前を付けます。
   - パブリッシャー設定で指定された証明書を、OCSP ユーザーアカウントのユーザー証明書として使用します。通常、これは CA のサブシステム証明書です。

   サブシステムユーザーの設定については、「ユーザーの作成」 で説明されています。

パブリッシャーを設定した後、「ルールの作成」 の説明に従って、発行された証明書と CRL のルールを設定します。



注記

pkiconsole が非推奨になりました。

9.4. LDAP ディレクトリーへの公開設定

公開を設定する一般的なプロセスには、証明書または CRL を特定の場所に公開するように発行者を設定することが含まれます。使用する場所の数に応じて、単一のパブリッシャーまたは複数のパブリッシャーが存在する可能性があります。場所は、証明書と CRL、または証明書の種類などのより細かい定義によって分割できます。ルールは、発行者に関連付けられることにより、発行するタイプと場所を決定します。

LDAP 公開の設定は、ディレクトリーを設定するための追加のステップを除き、他の公開手順と似ています。

1. 公開される証明書の Directory Server を設定します。特定の属性をエントリーに追加し、ID をバインドし、認証方法を設定する必要があります。
2. 公開された各オブジェクトのパブリッシャーを設定します (CA 証明書、クロスペア証明書、CRL、およびユーザー証明書)。パブリッシャーは、オブジェクトを格納する属性を宣言します。デフォルトで設定される属性は、各オブジェクトタイプを保存する X.500 標準属性です。この属性はパブリッシャーで変更できますが、通常は LDAP パブリッシャーを変更する必要はありません。
3. エントリーの DN が証明書のサブジェクト名から派生できるようにマッパーを設定します。通常、CA 証明書、CRL、およびユーザー証明書にを設定する必要はありません。証明書タイプに複数のマッパーを設定できます。これは、たとえば、ディレクトリーツリーの異なる部分にある会社の異なる部門の 2 組のユーザーの証明書を公開する場合に役立ちます。グループごとにマッパーが作成され、ツリーの異なるブランチを指定します。

   マッパーの設定に関する詳細は、「マッパーの作成」 を参照してください。

4. 「ルールの作成」 で説明されているように、パブリッシャーをマッパーに接続するルールを作成します。

5. 「公開の有効化」 の説明に従って、パブリッシュを有効にします。

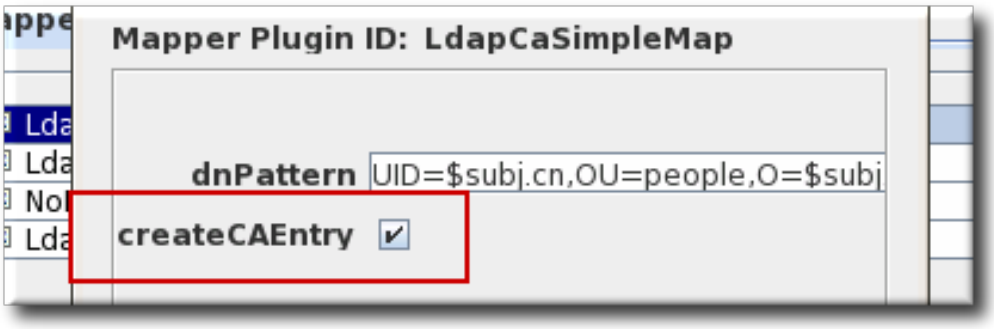
9.4.1. LDAP ディレクトリーの設定

証明書および CRL を公開する前に、Directory Server がパブリッシュシステムと連携するように設定する必要があります。つまり、ユーザーエントリーには証明書情報を受信できる属性が必要で、CRL を表すためにエントリーを作成する必要があります。

1. CA のエントリーを設定します。Certificate Manager が CA 証明書および CRL を公開するには、ディレクトリーには CA のエントリーが含まれている必要があります。

ヒント

LDAP 公開が設定されている場合、Certificate Manager はディレクトリー内の CA のエントリーを自動的に作成または変換します。このオプションは CA マッパーインスタンスおよび CRL マッパーインスタンスの両方で設定され、デフォルトで有効になっています。ディレクトリーが Certificate Manager がディレクトリーでエントリーを作成しないようにする場合は、これらのマッパーインスタンスでこのオプションを無効にし、CA を手動でディレクトリーに追加します。

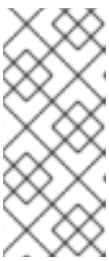


CA のエントリーをディレクトリーに追加する場合は、CA の DN に基づいてエントリータイプを選択します。

- CA の DN が **cn** コンポーネントで開始する場合は、CA の新規 **person** エントリーを作成します。別のタイプのエントリーを選択すると、**cn** コンポーネントが指定されない場合があります。

- CA の DN が **ou** コンポーネントで開始する場合は、CA の新規 **organizationalunit** エントリーを作成します。

このエントリーは、**pkiCA** または **certificationAuthority** オブジェクトクラスにはありません。証明書マネージャーは、このエントリーを CA の署名証明書を公開して、**pkiCA** または **certificationAuthority** オブジェクトクラスに自動的に変換します。



注記

pkiCA オブジェクトクラスは RFC 4523 に定義されています が、**certificationAuthority** オブジェクトクラスは (obsolete) RFC 2256 に定義されています。Directory Server が使用するスキーマ定義に応じて、いずれかのオブジェクトクラスは受け入れ可能です。場合によっては、両方のオブジェクトクラスを同じ CA エントリーに使用できます。

ディレクトリーエントリーの作成に関する詳細は、Red Hat Directory Server のドキュメントを参照してください。

2. CA およびユーザーディレクトリーエントリーに正しいスキーマ要素を追加します。

   Certificate Manager が証明書と CRL をディレクトリーに公開できるようにするには、特定の属性およびオブジェクトクラスで設定する必要があります。

オブジェクトタイプ	スキーマ	理由
エンドエンティティー証明書	userCertificate;binary (属性)	これは、証明書マネージャーが証明書をパブリッシュする属性です。 これは多値の属性で、各値は DER でエンコードされたバイナリー X.509 証明書です。**inetOrgPerson** という名前の LDAP オブジェクトクラスによりこの属性が許可されます。**strongAuthenticationUser** オブジェクトクラスはこの属性を許可し、他のオブジェクトクラスと組み合わせて、証明書を他のオブジェクトクラスのディレクトリーエントリーに公開できるようにすることができます。Certificate Manager は、このオブジェクトクラスを対応する Directory Server のスキーマテーブルに自動的に追加しません。 見つかったディレクトリーオブジェクトが **userCertificate;binary** 属性を許可しないと、証明書の追加または削除に失敗します。
CA 証明書	caCertificate;binary (属性)	これは、証明書マネージャーが証明書をパブリッシュする属性です。 Certificate Manager は、サーバーの起動時に独自の CA ディレクトリーエントリーに独自の CA 証明書を公開します。エントリーは、Certificate Manager の発行者名に対応します。 これは、**pkiCA** または **certificationAuthority** オブジェクトクラスの必須の属性です。Certificate Manager は、CA のディレクトリーエントリーを見つけると、このオブジェクトクラスを CA のディレクトリーエントリーに追加します。

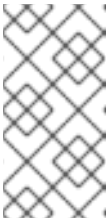
オブジェクトタイプ	スキーマ	理由
CRL	certificateRevocationList;binary (属性)	これは、Certificate Manager が CRL を公開する属性です。 Certificate Manager は、CRL を独自の LDAP ディレクトリーエントリーに公開します。エントリーは、Certificate Manager の発行者名に対応します。 これは、**pkiCA** または **certificationAuthority** オブジェクトクラスの属性です。属性の値は DER でエンコードされたバイナリー X.509 CRL です。CA のエントリーには、エントリーに CRL を公開するために、**pkiCA** または **certificationAuthority** オブジェクトクラスがすでに含まれている必要があります。
デルタ CRL	deltaRevocationList;binary (属性)	これは、Certificate Manager が証明書を公開する属性です。Certificate Manager は、フル CRL とは別に、デルタ CRL を独自の LDAP ディレクトリーエントリーに公開します。デルタ CRL エントリーは、Certificate Manager の発行者名に対応します。 この属性は、**deltaCRL** または **certificationAuthority-V2** オブジェクトクラスに属します。属性の値は DER でエンコードされたバイナリー X.509 delta CRL です。

3. Directory Server にアクセスするために使用する Certificate Manager のバインド DN を設定します。

   Certificate Manager は、証明書と CRL をディレクトリーに公開するために、ディレクトリーへの読み取り/書き込み権限を持っている必要があります。これにより、Certificate Manager は、証明書関連情報を含むユーザーエントリーと、CA の証明書および CRL 関連情報を含む CA エントリーを変更できます。

   バインド DN エントリーは、以下のいずれかになります。

   - Directory Manager などの書き込みアクセスを持つ既存の DN。
   - 書き込みアクセスが付与された新規ユーザー。エントリーは、Certificate Manager の DN で識別することができます (例: **cn=testCA, ou=Research Dept, o=Example Corporation, st=California, c=US**)。



注記

このユーザーに付与される特権を慎重に検討してください。このユーザーは、アカウントの ACL を作成して、そのディレクトリーに書き込みできます。Certificate Manager のエントリーへの書き込みアクセス権限を付与する方法については、Directory Server のドキュメントを参照してください。

4. Certificate Manager が Directory Server に対して認証する方法のディレクトリー認証方法を設定します。Basic 認証 (簡易ユーザー名およびパスワード)、クライアント認証なしの SSL、およびクライアント認証を使用する SSL (証明書ベース) の 3 つのオプションがあります。

   サーバーとの通信方法の設定は、Red Hat Directory Server のドキュメントを参照してください。

9.4.2. LDAP パブリッシャーの設定

Certificate Manager は、LDAP 公開に関連するパブリッシャーのセットを作成、設定、および有効にします。デフォルトのパブリッシャー (CA 証明書、ユーザー証明書、CRL、およびクロスのペア証明書用) は、証明書および CRL を保存するための X.500 標準属性に準拠しており、変更する必要はありません。

表9.1 LDAP パブリッシャー

パブリッシャー	説明
LdapCaCertPublisher	CA 証明書を LDAP ディレクトリーに公開します。
LdapCrlPublisher	CRL を LDAP ディレクトリーに公開します。
LdapDeltaCrlPublisher	デルタ CRL を LDAP ディレクトリーに公開します。
LdapUserCertPublisher	すべての種類のエンドエンティティー証明書を LDAP ディレクトリーに公開します。
LdapCrossCertPairPublisher	相互署名付き証明書を LDAP ディレクトリーに公開します。

9.4.3. マッパーの作成

マッパーは LDAP 公開のみで使用されます。マッパーは、証明書のサブジェクト名と、証明書が公開されるディレクトリーエントリーの DN 間の関係を定義します。Certificate Manager は、使用するエントリーを判断できるように、証明書または証明書要求からエントリーの DN を取得する必要があります。マッパーは、ユーザーエントリーの DN と証明書のサブジェクト名またはその他の入力情報との関係を定義して、エントリーの正確な DN を特定し、ディレクトリーで見つけることができるようにします。

設定すると、Certificate Manager は、最も一般的な関係を定義するマッパーのセットを自動的に作成します。デフォルトのマッパーは、表9.2「デフォルトのマッパー」 に一覧表示されます。

表9.2 デフォルトのマッパー

マッパー	説明
LdapUserCertMap	ユーザー証明書を公開するために、ディレクトリーでユーザーエントリーの正しい属性を見つけます。
LdapCrlMap	CRL を公開するために、ディレクトリーで CA のエントリーの正しい属性を見つけます。
LdapCaCertMap	CA 証明書を公開するために、ディレクトリーで CA のエントリーの正しい属性を見つけます。

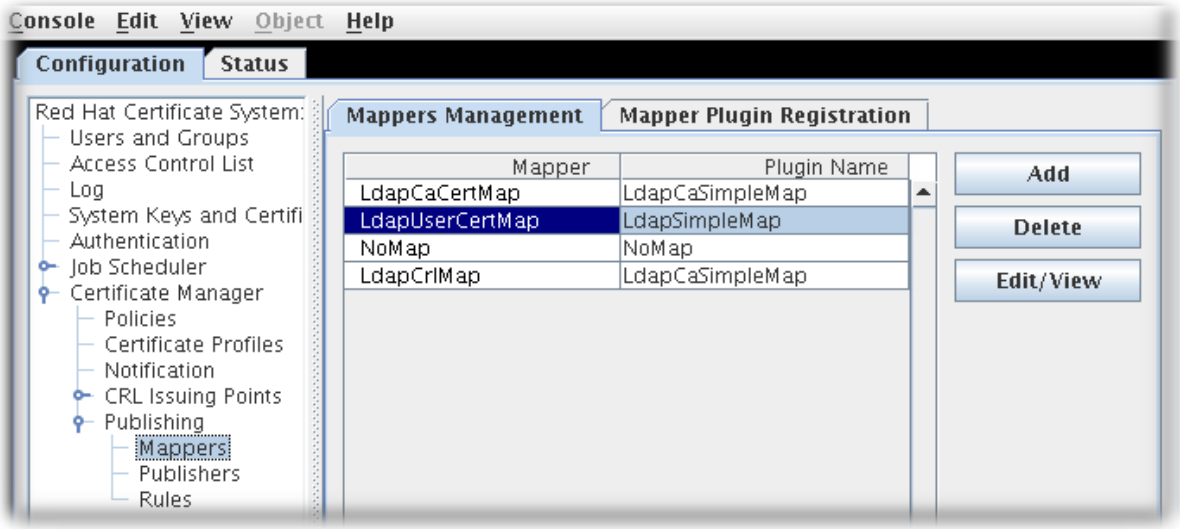
デフォルトのマッパーを使用するには、DN パターンを指定し、ディレクトリーに CA エントリーを作成するかどうかを指定して、各マクロを設定します。他のマッパーを使用するには、マッパーのインスタンスを作成および設定します。詳細は、「マッパープラグインモジュール」を参照してください。

1. Certificate Manager コンソールにログインします。

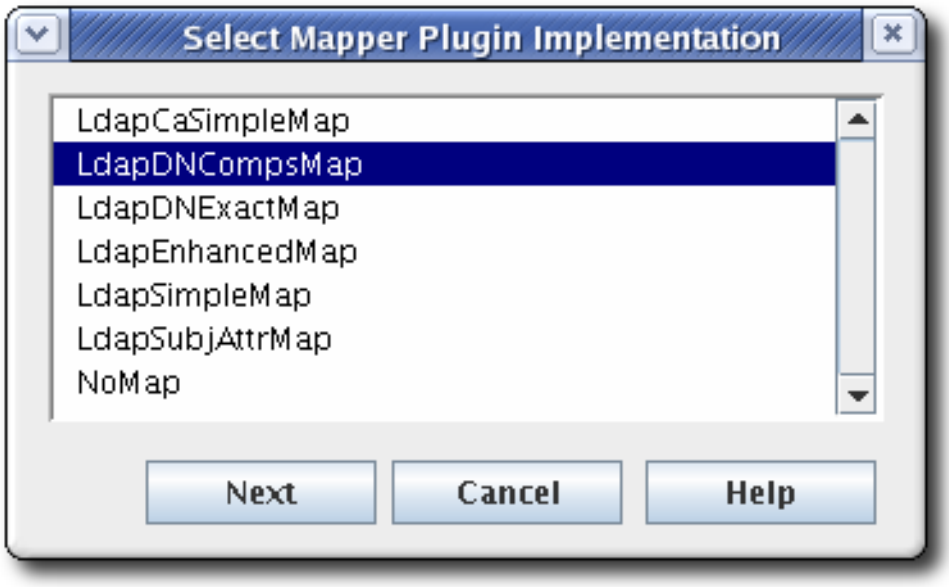
   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Mappers** を選択します。

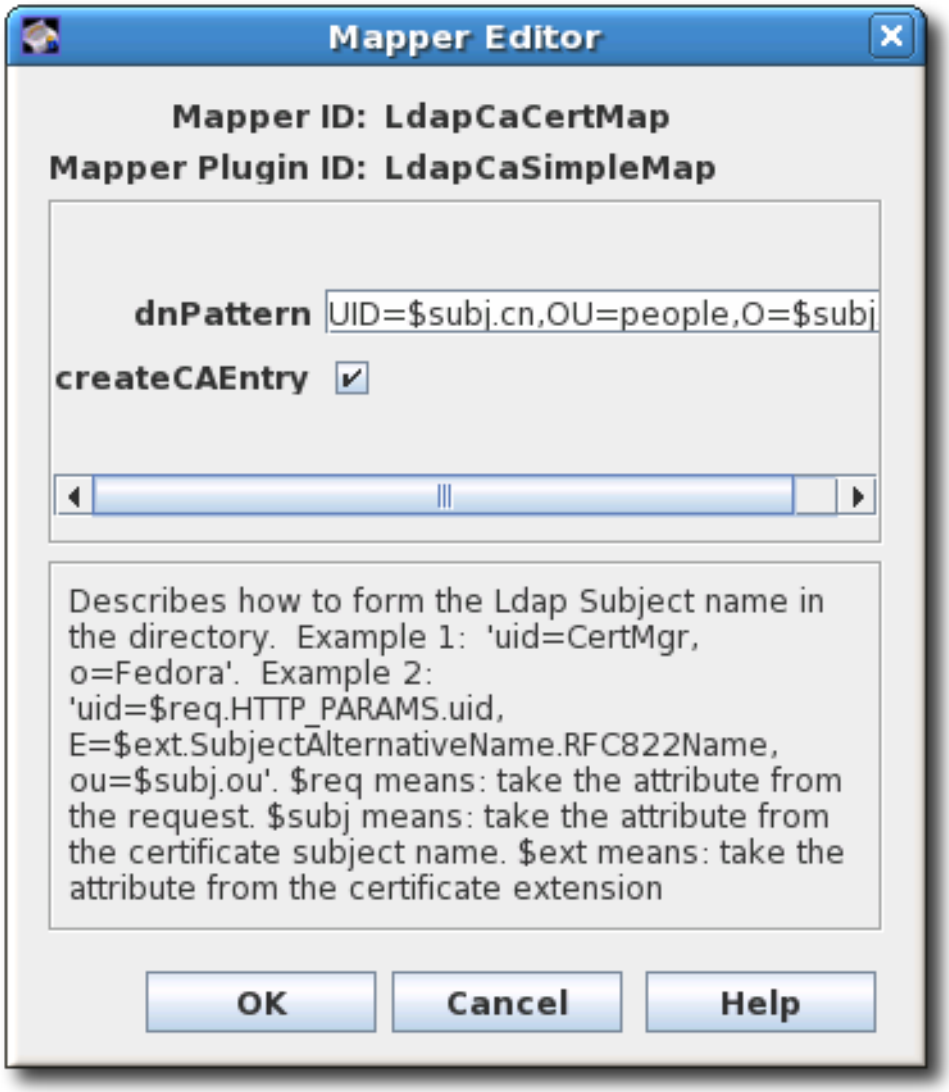
   設定されたマッパーを一覧表示する **Mappers Management** タブが右側で開きます。



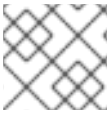
3. 新しいマッパーインスタンスを作成するには、**Add** をクリックします。**Select Mapper Plugin Implementation** ウィンドウが開き、登録されたマッパーモジュールが一覧表示されます。モジュールを選択し、そのモジュールを編集します。これらのモジュールに関する詳細は、「マッパープラグインモジュール」を参照してください。



4. マッパーインスタンスを編集し、**OK** をクリックします。



各マッパーに関する詳細は、「マッパープラグインモジュール」 を参照してください。



注記

pkiconsole が非推奨になりました。

9.4.4. 設定の完了: ルールおよび有効化

LDAP 公開のマッパーを設定したら、「ルールの作成」 の説明に従って、公開証明書および CRL のルールを設定します。

設定が完了したら、「公開の有効化」 の説明に従って公開を有効にします。

9.5. ルールの作成

ルールは、どの場所にどの証明書オブジェクトを公開するかを決定します。ルールは、連携してではなく、独立して機能します。公開される証明書または CRL は、すべてのルールに対して照合されます。一致するルールはすべてアクティブになります。このようにして、ファイルベースのルール、OCSP ルール、およびディレクトリーベースのルールを照合することにより、同じ証明書または CRL をファイル、Online Certificate Status Manager、および LDAP ディレクトリーに公開できます。

ルールは、各オブジェクトタイプ (CA 証明書、CRL、ユーザー証明書、およびクロスペアの証明書) に設定できます。ルールは、さまざまな種類の証明書またはさまざまな種類の CRL についてより詳細にすることができます。

ルールは最初に、ルールに設定されたタイプと述語をオブジェクトと照合することにより、オブジェクトが一致するかどうかを判断します。一致するオブジェクトは、ルールに関連付けられたパブリッシャーとマッパーにより公開されます。

Certificate Manager が発行する証明書のタイプごとにルールが作成されます。

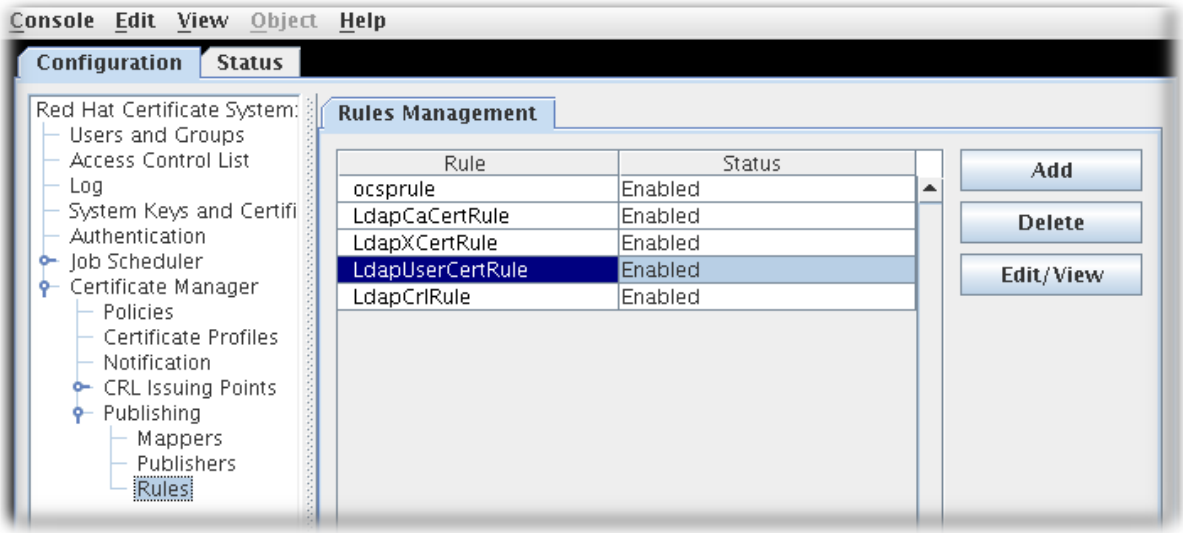
次の手順を実行して、公開ルールを変更します。

1. Certificate Manager コンソールにログインします。

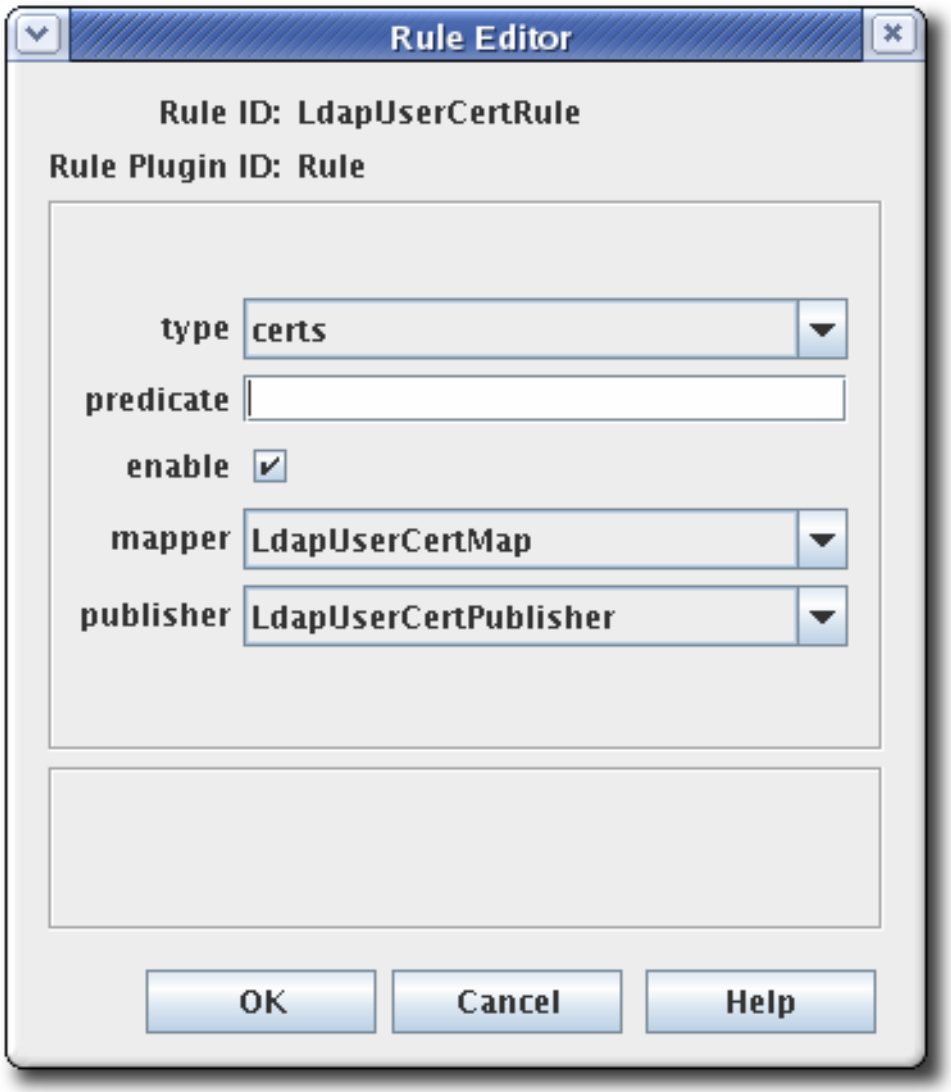
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択し、**Rules** を選択します。

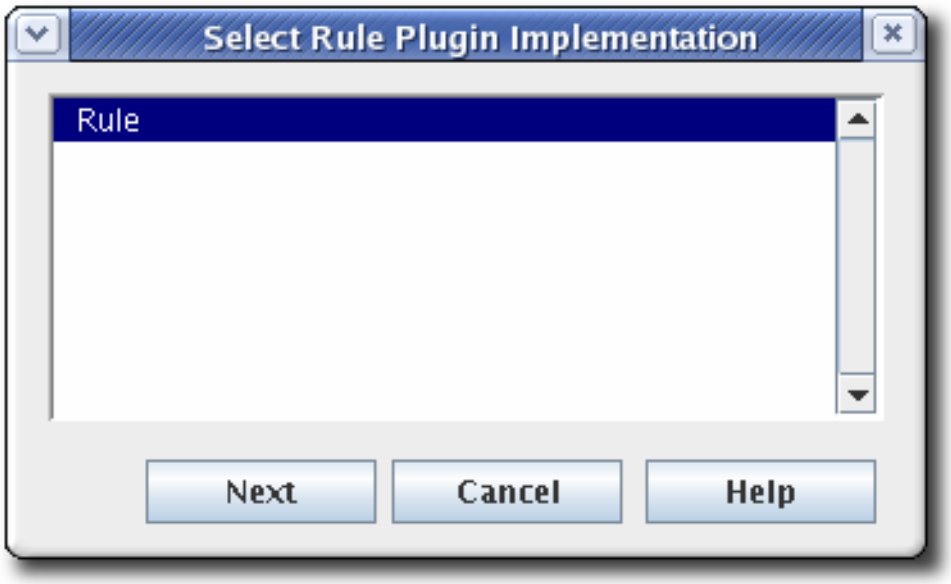
   設定したルールを一覧表示する **Rules Management** タブが右側で開きます。



3. 既存のルールを編集するには、一覧からそのルールを選択して、**Edit** をクリックします。これにより、**Rule Editor** ウィンドウが開きます。

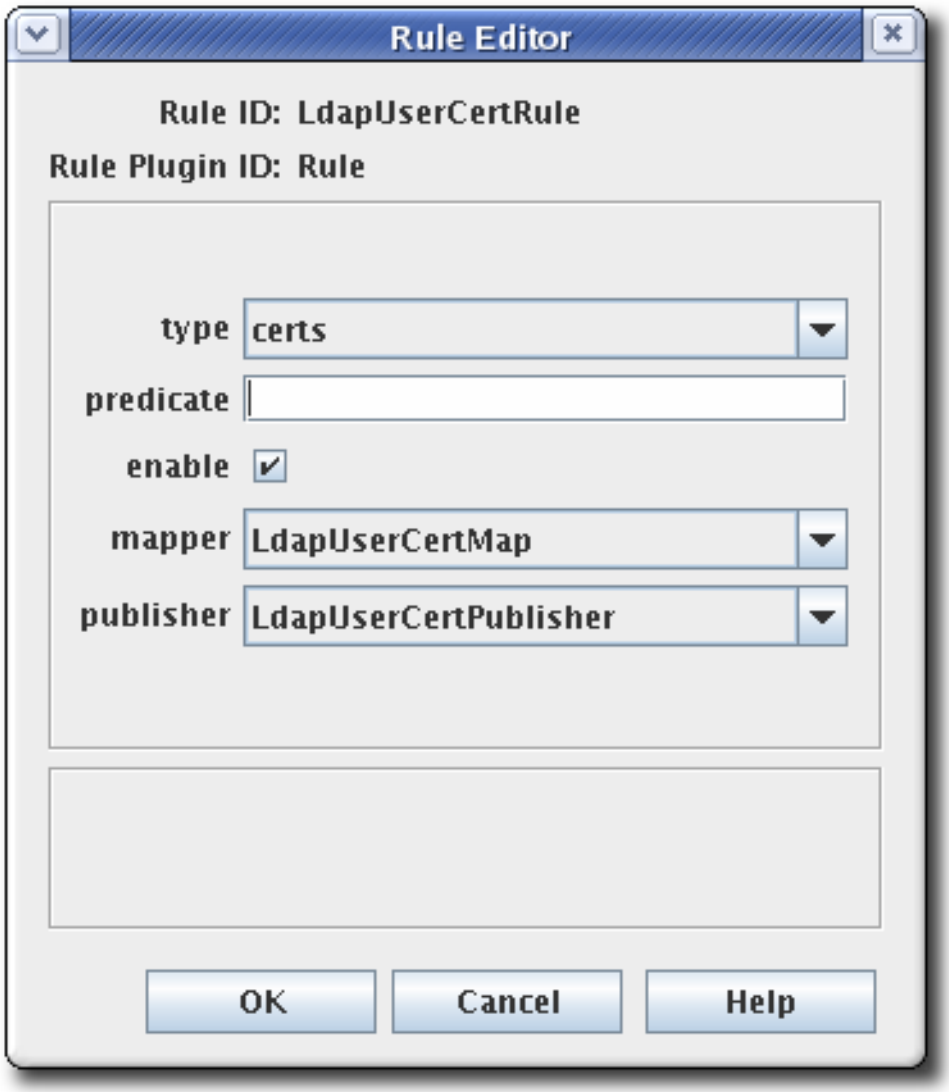


4. ルールを作成するには、**Add** をクリックします。これにより、**Select Rule Plug-in Implementation** ウィンドウが開きます。



Rule モジュールを選択します。これは唯一のデフォルトモジュールです。カスタムモジュールが登録されている場合は、それらも利用できます。

5. ルールを編集します。



- **type**。これは、ルールが適用される証明書のタイプです。CA 署名証明書の場合、値は **cacert** です。自己署名証明書の場合、値は **xcert** です。その他すべてのタイプの証明書の場合、値は **certs** です。CRL には **crl** を指定します。
- **predicate**。これにより、このルールが適用される証明書または CRL 発行ポイントのタイプに述語の値を設定します。CRL 発行ポイント、デルタ CRL、および証明書の述語値の一覧は 表9.3「述語式」 に表示されます。
- **enable**。
- **mapper**。ファイルに公開する場合、マッパーは必要ありません。LDAP 公開にのみ必要です。このルールが LDAP ディレクトリーに公開されるパブリッシャーに関連付けられている場合は、ここに適切なマッパーを選択します。他のすべての形式の発行では空白のままにします。
- **publisher**。ルールに関連付けるパブリッシャーを設定します。

表9.3「述語式」 は、CRL 発行ポイントおよびデルタ CRL および証明書プロファイルを識別するために使用できる述語を一覧表示します。

表9.3 述語式

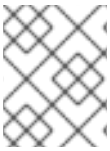
述語タイプ	述語
CRL 発行ポイント	**issuingPointId==**Issuing_Point_Instance_ID **&& isDeltaCRL==[true|false]** マスター CRL のみを公開するには、**isDeltaCRl==false** を設定します。デルタ CRL のみを公開するには、**isDeltaCRL==true** を設定します。両方を公開するには、マスター CRL のルールとデルタ CRL の別のルールを設定します。
証明書プロファイル	**profileId==**profile_name 使用するプロファイルに基づいて証明書を公開するには、**profileId==** を **caServerCert** などのプロファイル名に設定します。

注記

pkiconsole が非推奨になりました。

9.6. 公開の有効化

公開は、ファイル、LDAP、またはその両方に対して有効にできます。パブリッシャー、ルール、およびマッパーの設定後に公開を有効にする必要があります。有効にすると、サーバーは公開を開始しようとします。公開が有効になる前に正しく設定されていなかった場合、公開は望ましくない動作を示したり、失敗したりする可能性があります。



注記

CRL を設定します。CRL は公開前に設定する必要があります。*7章 証明書の取り消しおよびCRL 発行* を参照してください。

1. Certificate Manager コンソールにログインします。

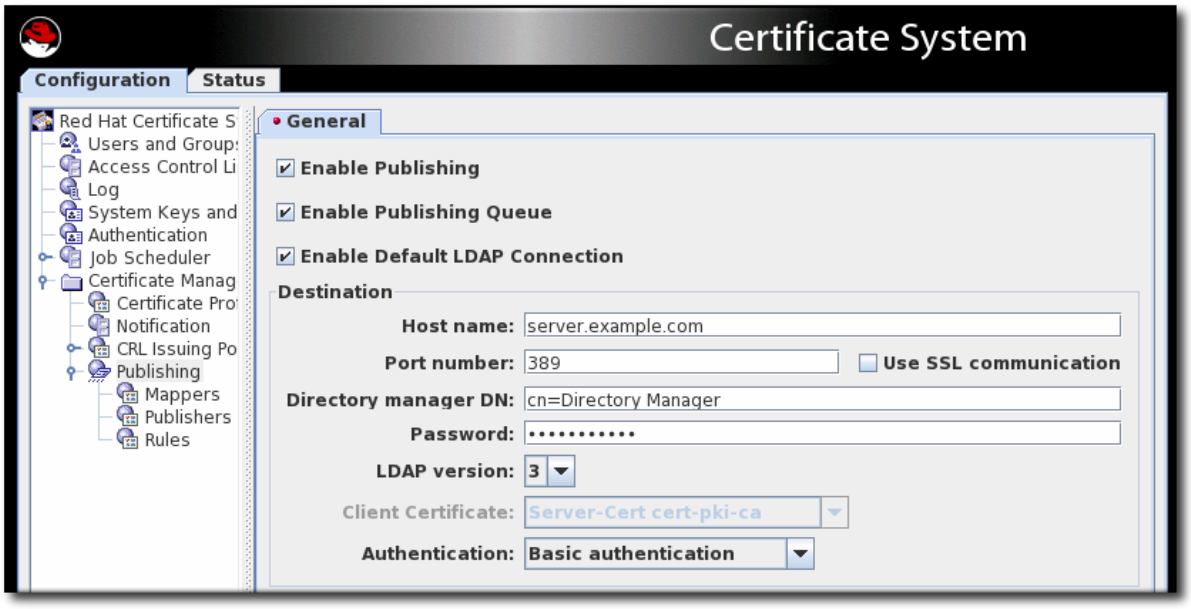
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択します。

   右側のペインには、LDAP 準拠のディレクトリーに公開するための詳細情報が表示されます。

3. ファイルの公開のみを有効にするには、**Enable Publishing** を選択します。

4. LDAP 公開を有効にするには、**Enable Publishing** および **Enable Default LDAP Connection** の両方を選択します。



Destination セクションで、Directory Server インスタンスの情報を設定します。

- **Host name**。Directory Server が SSL クライアント認証通信用に設定されている場合、名前は Directory Server の SSL サーバー証明書のサブジェクト DN 内の **cn** コンポーネントに一致する必要があります。

  ホスト名は完全修飾ドメイン名または IPv4 アドレスまたは IPv6 アドレスになります。

- **Port number**。

- **Directory Manager DN**これは、Directory Manager 権限を持つディレクトリーエントリーの識別名 (DN) です。Certificate Manager はこの DN を使用してディレクトリーツリーにアクセスし、そのディレクトリーに公開します。この DN に設定されたアクセス制御は、Certificate Manager が公開できるかどうかを判断します。公開システムが実際に書き込む必要のある属性に対してのみ読み取り/書き込み権限が制限されている別の DN を作成することができます。

- **Password**。これは、CA が証明書または CRL の公開先の LDAP ディレクトリーにバインドするために使用するパスワードです。Certificate Manager は、このパスワードを **password.conf** ファイルに保存します。以下に例を示します。

  ```
  CA LDAP Publishing:password
  ```

  注記

  パブリッシュパスワード (**CA LDAP Publishing**) を識別するパラメーター名は、**ca.publish.ldappublish.ldap.ldapauth.bindPWPrompt** パラメーターの Certificate Manager の **CS.cfg** ファイルで設定されます。

- **クライアント証明書**。これにより、SSL クライアント認証に Certificate Manager が使用する証明書がパブリッシュディレクトリーに設定されます。デフォルトでは、証明書マネージャーは SSL サーバー証明書を使用します。

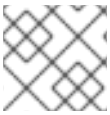
- **LDAP バージョン**。LDAP バージョン 3 を選択します。

- **Authentication**。Certificate Manager が Directory Server に対して認証する方法。 **Basic authentication** と **SSL client authentication** を選択できます。

Directory Server が基本認証またはクライアント認証なしの SSL 通信用に設定されている場合は、**Basic authentication** を選択して、Directory マネージャーの DN とパスワードの値を指定します。

Directory Server がクライアント認証を使用した SSL 通信用に設定されている場合は、**SSL client authentication** と **Use SSL communication** オプションを選択して、Certificate Manager がディレクトリーへの SSL クライアント認証に使用する必要のある証明書を識別します。

サーバーは、Directory Server への接続を試みます。情報が正しくない場合、サーバーにはエラーメッセージが表示されます。



注記

pkiconsole が非推奨になりました。

9.7. 公開キューの有効化

登録プロセスには、発行した証明書をディレクトリーまたはファイルに公開します。したがって、基本的には、最初の証明書要求を閉じます。ただし、外部ネットワークに証明書を公開すると、発行プロセスが大幅に遅くなり、リクエストが開いたままになります。

この状況を回避するために、管理者は **公開キュー** を有効にできます。パブリッシュキューは、別のリクエストキューを使用するリクエスト操作および登録操作 (外部の LDAP ディレクトリーを伴う可能性がある) を分離します。要求キューはすぐに更新され、登録プロセスが完了したことを示します。一方、公開キューがネットワークトラフィックの一時停止時に情報を送信します。

公開キューは、承認された証明書ごとに新しいスレッドを開くのではなく、生成された証明書を公開する定義済みの制限された数のスレッドを設定します。

パブリッシュキューはデフォルトで無効になっています。公開を有効にするとともに、CA コンソールで有効にすることができます。

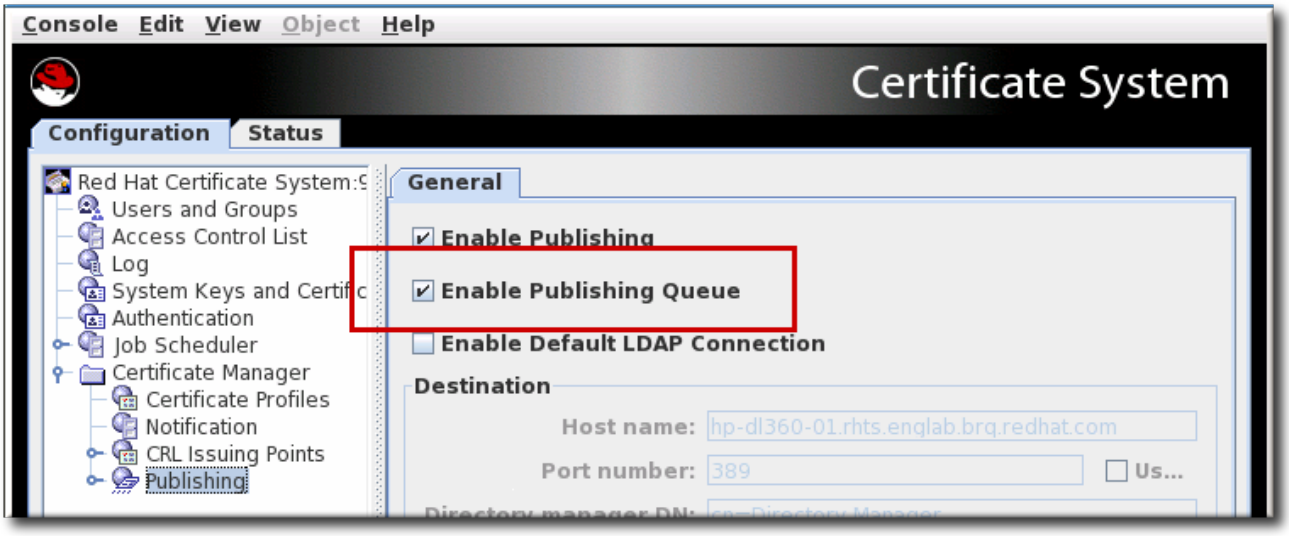
注記

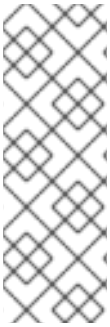
pkiconsole が非推奨になりました。

注記

パブリッシュキューはデフォルトで無効になっていますが、**コンソールで** LDAP 公開が有効な場合にはキューが自動的に有効になります。それ以外の場合は、キューを手動で有効にできます。

図9.1 公開キューの有効化





ヒント

CS.cfg ファイルを編集してパブリッシュキューを有効にすると、管理者はパブリッシュ操作に使用するスレッドの数やキューページサイズなど、パブリッシュする他のオプションを設定できます。

CS.cfg ファイルを編集してこの機能を設定する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『公開キューの有効化および設定』 セクションを参照してください。

9.8. 再開可能な CRL ダウンロードの設定

Certificate System は、中断された CRL ダウンロードをスムーズに再開するオプションを提供します。これは、HTTP 経由でプレーンファイルとして CRL を公開することで行われます。CRL のダウンロード方法により、CRL の取得やネットワーク全体の輻輳を軽減することができます。

9.8.1. wget を使用した CRL の取得

CRL は HTTP 経由でテキストファイルとして公開されるため、**wget** などのツールを使用して CA から手動で取得できます。**wget** コマンドを使用すると、公開されている CRL を検索できます。たとえば、以前のフル CRL よりも新しいフル CRL を取得するには、次のようにします。

```
[root@server ~]# wget --no-check-certificate -d
https://server.example.com:8443/ca/ee/ca/crl/MasterCRL.bin
```

wget の関連パラメーターの概要は、表9.4「CRL の取得に使用する wget オプション」 にまとめています。

表9.4 CRL の取得に使用する wget オプション

引数	説明
引数なし	完全な CRL を取得します。

引数	説明
-N	ローカルコピー (delta CRL) よりも新しい CRL を取得します。
-c	部分的にダウンロードしたファイルを取得します。
--no-check-certificate	接続の SSL を省略するため、ホストとクライアント間で SSL を設定する必要はありません。
-d	デバッグ情報を出力します。

9.9. ペア間の証明書の公開

クロスペアの証明書は LDAP ディレクトリーまたはファイルに **crossCertificatePair** エントリーとして公開できます。これはデフォルトで有効になっています。これが無効な場合は、Certificate Manager Console で以下のコマンドを実行して再度有効にできます。

1. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、左側のペインの **Certificate Manager** リンクを選択してから、**Publishing** リンクを選択します。
3. **Publishing** の **Rules** リンクをクリックし ます。これにより、右側に **Rules Management** ペインが開きます。
4. ルールが存在し、無効になっている場合は、**enable** チェックボックスを選択します。ルールが削除された場合は、**Add** クリックして新しいルールを作成します。
    a. **type** ドロップダウンメニューから **xcerts** を選択します。
    b. **有効** のチェックボックスが選択されていることを確認してください。
    c. **mapper** ドロップダウンメニューから、**LdapCaCertMap** を選択します。
    d. **publisher** ドロップダウンメニューから **LdapCrossCertPairPublisher** を選択します。

公開ルールで指定されたマッパーとパブリッシャーは両方とも、CA コンソールの左側のナビゲーションウィンドウの **Publishing** リンクの下の **Mapper** と **Publisher** 下に一覧表示されます。デフォルトでは、マッパーの **LdapCaCertMap** は、**LdapCaSimpleMap LDAP** エントリーに **crossCertificatePair** を保存するように指定します。パブリッシャー **LDAPCrossPairPublisher** はデフォルトで、CA エントリーにクロスペア証明書を **crossCertificatePair;binary** に保存する属性を設定します。

ペア間の証明書の使用に関する詳細は、「ペア間の証明書の使用」 を参照してください。

クロスペア証明書プロファイルの作成に関する詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『クロスペアプロファイルの設定』 セクションを参照してください。

注記

pkiconsole が非推奨になりました。

9.10. ファイルへの公開テスト

Certificate Manager が証明書と CRL を正しくファイルに正常に公開していることを確認するには、以下を実行します。

1. CA のエンドエンティティーを開き、証明書をリクエストします。
2. 必要に応じて、エージェントサービスページを使用してリクエストを承認します。
3. エンドエンティティーページから証明書を取得し、証明書をブラウザーにダウンロードします。
4. サーバーが、証明書を含む DER でエンコードされたファイルを生成したかどうかを確認します。

   証明書のバイナリーブロブが公開されることになっているディレクトリーを開きます。証明書ファイルの名前は **cert-serial_number.der** にする必要があります。
5. Binary to ASCII ツールを使用して、DER でエンコードされた証明書をベース 64 でエンコードされた形式に変換します。このツールの詳細は、**BtoA(1)** man ページを参照してください。

   ```
   BtoA input_file output_file
   ```

   input_file は、DER でエンコードされた証明書が含まれるファイルへのパスを設定し、**output_file** は、base-64 でエンコードされた証明書を書き込むようにファイルへのパスを設定します。
6. ASCII ファイルを開きます。base-64 でエンコードされた証明書は以下のように表示されます。

   ```
   -----BEGIN CERTIFICATE-----
   MMIIBtgYJYIZIAYb4QgIFoIIBpzCCAZ8wggGbMIIBRaADAgEAAgEBMA0GCSqGSIb3DQEBB
   AUAMFcxC
   AJBgNVBAYTAlVTMSwwKgYDVQQKEyNOZXRzY2FwZSBDb21tdW5pY2F0aWhfyyuougjgjjg
   mkgjkgmjg
   fjfgjjjgfyjfyj9ucyBDb3Jwb3JhdGlvbjpMEaMBgGA1UECxMRSXNzdWluZyhgdfhbfdpffjphotoo
   gdhkBBdXRob3JpdHkwHhcNOTYxMTA4MDkwNzM0WhcNOTgxMTA4MDkwNzMM0WjBXMQ
   swCQYDVQQGEwJ
   VUzEsMCoGA1UEChMjTmV0c2NhcGUgQ29tbXVuaWNhdGlvbnMgQ29ycG9yY2F0aW9ucyB
   Db3Jwb3Jhd
   GlvbjpMEaMBgGA1UECxMRSXNzdWluZyBBdXRob3JpdHkwHh
   -----END CERTIFICATE-----
   ```
7. Pretty Print Certificate ツールを使用して、ベース 64 でエンコードされた証明書を読み取り可能なフォームに変換します。このツールの詳細は、**PrettyPrintCert(1)** man ページを参照してください。

   ```
   PrettyPrintCert input_file [output_file]
   ```

input_file は base-64 でエンコードされた証明書が含まれる ASCII ファイルへのパスを設定し、任意で **output_file** に証明書を書き込むファイルにパスを設定できます。出力ファイルが設定されていない場合、証明書情報は標準出力に書き込まれます。

8. 出力と、発行された証明書を比較します。証明書のシリアル番号と、ファイル名で使用されている証明書と比較します。

   すべてが一致する場合、Certificate Manager は証明書をファイルに公開するように正しく設定されています。

9. 証明書を取り消します。

10. サーバーが、CRL を含む DER でエンコードされたファイルを生成したかどうかを確認します。

    サーバーが CRL をバイナリーブロブとして公開するディレクトリーを開きます。CRL ファイルの名前は **crl-this_update.der** であるはずです。**this_update** は、CRL の時間依存の **This Update** 変数から派生した値を指定します。

11. Binary to ASCII ツールを使用して、DER でエンコードされた CRL をベース 64 でエンコードされた形式に変換します。

    ```
    BtoA input_file output_file
    ```

12. Pretty Print CRL ツールを使用して、base 64 でエンコードされた CRL を読み取り可能なフォームに変換します。

    ```
    PrettyPrintCrl input_file [output_file]
    ```

13. 出力を比較します。

9.11. ファイルに公開される証明書および CRL の表示

証明書と CRL は、base-64 でエンコードされたファイルまたは DER エンコードの 2 種類のファイルに公開できます。これらのファイルの内容は、**dumpasn1** ツールまたは **PrettyPrintCert** または **PrettyPrintCrl** ツールを使って Pretty-print 形式にこのファイルを変換して表示できます。

base-64 でエンコードされたファイルのコンテンツを表示するには、以下を実行します。

1. base-64 ファイルをバイナリーに変換します。以下に例を示します。

   ```
   AtoB /tmp/example.b64 /tmp/example.bin
   ```

2. **PrettyPrintCert** または **PrettyPrintCrl** ツールを使用して、バイナリーファイルを pretty-print 形式に変換します。以下に例を示します。

   ```
   PrettyPrintCert example.bin example.cert
   ```

DER でエンコードされたファイルの内容を表示するには、DER エンコードファイルで **dumpasn1**、**PrettyPrintCert**、または **PrettyPrintCrl** ツールを実行するだけです。以下に例を示します。

```
PrettyPrintCrl example.der example.crl
```

9.12. ディレクトリーの証明書および CRL の更新

Directory Server がダウンしているときに証明書が発行または取り消されると、Certificate Manager と公開ディレクトリーが同期しなくなる可能性があります。発行または失効した証明書は、Directory Server のバックアップ時に手動で公開または公開解除する必要があります。

ディレクトリーと同期していない証明書 (ディレクトリーにない有効な証明書と、ディレクトリーに残っている失効または期限切れの証明書) を見つけるために、Certificate Manager は、内部データベース内の証明書がディレクトリーに公開されているかどうかの記録を保持します。Certificate Manager および公開ディレクトリーの同期がなくなる場合は、Certificate Manager エージェントサービスページの **Update Directory** オプションを使用して、公開ディレクトリーを内部データベースと同期します。

ディレクトリーと内部データベースの同期には、以下の選択肢を利用できます。

- 内部データベースで同期していない証明書を検索し、公開または非公開にします。
- Directory Server のダウン中に発行された証明書を公開します。同様に、Directory Server の停止時に取り消された、または有効期限が切れた証明書も使用できます。
- シリアル番号 **xx** からシリアル番号 **yy** までのシリアル番号に基づいて、一連の証明書を公開または非公開にします。

Certificate Manager の公開ディレクトリーは、Certificate Manager エージェントからのみ手動で更新できます。

9.12.1. ディレクトリーでの証明書の手動による更新

Certificate Manager エージェントサービスページの **Update Directory Server** フォームを使用すると、証明書関連の情報を使用してディレクトリーを手動で更新できます。このフォームは、以下の操作の組み合わせを開始します。

- 証明書でディレクトリーを更新します。
- 期限切れの証明書をディレクトリーから削除します。

  自動化されたジョブをスケジュールすることにより、公開ディレクトリーからの期限切れの証明書の削除を自動化できます。詳細は、13章 *自動ジョブの設定* を参照してください。
- ディレクトリーから失効した証明書を削除します。

以下のコマンドを実行して、ディレクトリーを変更で手動で更新します。

1. Certificate Manager エージェントサービスページを開きます。
2. **Update Directory Server** のリンクを選択します。
3. 適切なオプションを選択し、**Update Directory** をクリックします。

   Certificate Manager は、内部データベースの証明書情報でディレクトリーの更新を開始します。大幅に変更した場合は、ディレクトリーの更新にかなりの時間がかかる可能性があります。この期間中、発行された証明書や取り消された証明書など、Certificate Manager を介して行われた変更は、更新に含まれない場合があります。ディレクトリーの更新中に証明書が発行または取り消された場合は、それらの変更を反映するようにディレクトリーを再度更新してください。

ディレクトリーの更新が完了すると、Certificate Manager にステータスレポートが表示されます。プロセスが中断された場合、サーバーはエラーメッセージを記録します。

Certificate Manager がルート CA としてインストールされている場合、エージェントインターフェイスを使用してディレクトリーを有効な証明書で更新するときに、ユーザー証明書用に設定された公開ルールを使用して CA 署名証明書が公開されることがあります。これにより、オブジェクトクラスの違反エラーや他のマッパーの他のエラーが返される場合があります。CA 署名証明書を除外するために適切なシリアル番号範囲を選択すると、この問題を回避できます。CA 署名証明書は、ルート CA の最初の証明書です。

- **predicate** パラメーターの値を **profileId!=caCACert** に変更して、ユーザー証明書のデフォルト公開ルールを変更します。
- **LdapCaCertPublisher** プラグインモジュールを使用して別のルールを追加し、predicate パラメーターを **profileId=caCACert** に設定して CA 証明書を公開します。

9.12.2. ディレクトリーでの CRL の手動による更新

Certificate Manager エージェントサービスページの **Certificate Revocation List** フォームは、CRL 関連の情報のあるディレクトリーを手動で更新します。

以下のコマンドを実行して CRL 情報を手動で更新します。

1. Certificate Manager エージェントサービスページを開きます。
2. **Update Revocation List** を選択します。
3. **Update** をクリックします。

Certificate Manager は、内部データベースの CRL でディレクトリーの更新を開始します。CRL のサイズが大きい場合は、ディレクトリーの更新にかなり時間がかかります。この期間中、CRL への変更は更新に含まれない可能性があります。

ディレクトリーを更新すると、Certificate Manager にステータスレポートが表示されます。プロセスが中断された場合、サーバーはエラーメッセージを記録します。

9.13. カスタムマッパーの登録およびプラグインモジュールの公開

新しいマッパーまたはパブリッシャープラグインモジュールは、Certificate Manager のパブリッシングフレームワークに登録できます。不要なマッパーまたはパブリッシャーモジュールを削除できます。モジュールを削除する前に、このモジュールに基づくすべてのルールを削除します。

1. カスタムジョブクラスを作成します。この例では、カスタムパブリッシャープラグインは **MyPublisher.java** になります。
2. 新しいクラスをコンパイルします。

```
javac -d . -classpath $CLASSPATH MyPublisher.java
```

3. CA がカスタムクラスにアクセスできるように、CA の **WEB-INF** Web ディレクトリーにディレクトリーを作成します。

```
mkdir /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

4. 新しいプラグインファイルを新しい **class** ディレクトリーにコピーし、所有者を Certificate System system user (**pkiuser**) に設定します。

```
cp -pr com /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes

chown -R pkiuser:pkiuser /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
```

5. プラグインを登録します。

   a. Certificate Manager コンソールにログインします。

      ```
      pkiconsole https://server.example.com:8443/ca
      ```

   b. **Configuration** タブで、左側のナビゲーションツリーから **Certificate Manager** を選択します。**Publishing** を選択します。

   c. マッパーモジュールを登録するには、**Mappers** を選択し、**Mapper Plugin Registration** タブを選択します。

      パブリッシャーモジュールを登録するには、**Publishers** を選択し、**Publisher Plug-in Registration** タブを選択します。

   d. プラグインを登録するには、**Register** をクリックします。

   e. プラグイン名とプラグインクラス名を設定します。クラス名 (実装された Java クラスへのパス)このクラスがパッケージに含まれる場合は、パッケージ名を含めます。たとえば、**com.customplugins** という名前のパッケージに **customMapper** という名前のクラスを登録するには、名前は **com.customplugins.customMapper** になります。

注記

pkiconsole が非推奨になりました。

第10章 証明書を登録するための認証

本章では、エンドエンティティー証明書を登録する方法、サーバー証明書を作成および管理する方法、エンドエンティティー証明書を登録するときに使用する Certificate System で使用可能な認証方法、およびそれらの認証方法を設定する方法を説明します。

登録 は、エンドツーエンティティーに証明書を発行するプロセスです。このプロセスでは、リクエストを作成して送信し、それを要求しているユーザーを認証してから、リクエストを承認して証明書を発行します。

エンドエンティティーの認証に使用されるメソッドは、登録プロセス全体を決定します。Certificate System がエンティティーを認証できる方法は 3 つあります。

- **エージェントの承認** 登録では、承認のためにエンドエンティティーがエージェントに送信されます。エージェントは証明書要求を承認します。
- **自動** 登録では、エンドエンティティー要求はプラグインを使用して認証され、証明書要求が処理されます。エージェントは登録プロセスに関与していません。
- **CMC 登録** では、サードパーティーのアプリケーションが、エージェントによって署名されてから自動的に処理される要求を作成できます。

Certificate Manager は、最初にエージェント承認の登録と CMC 認証用に設定されています。自動登録を有効にするには、認証プラグインモジュールのいずれかを設定します。サブシステムの単一インスタンスで、1 つ以上の複数の認証方法を設定できます。

注記

自動通知を設定することにより、任意の認証方法で証明書が発行されたときに、電子メールをエンドエンティティーに自動的に送信できます。通知の詳細は、12章 *自動通知の使用* を参照してください。

10.1. エージェント承認登録の設定

Certificate Manager は、最初にエージェント承認の登録に対して設定されます。エンドエンティティーは、エージェントの承認のエージェントキューに送信されるリクエストを作成します。エージェントはリクエストを変更したり、リクエストのステータスを変更したり、リクエストを拒否したり、リクエストを承認することができます。リクエストが承認されると、署名済み要求が Certificate Manager に送信され、処理します。Certificate Manager はリクエストを処理し、証明書を発行します。

エージェントが承認した登録方法は設定できません。Certificate Manager が他の登録方法用に設定されていない場合、サーバーは、待機エージェントの承認先のキューに、証明書関連の要求をすべて自動的に送信します。これにより、認証情報がないすべてのリクエストが、エージェントの承認のために要求キューに送信されるようになります。

エージェントの承認登録を使用するには、プロファイルの **.cfg** ファイルに認証方法を空白のままにしておきます。以下に例を示します。

```
auth.instance_id=
```

10.2. 自動登録

自動登録では、ユーザーが認証プラグインモジュールで設定された方法で正常に認証されるとすぐに、エンドエンティティー登録要求が処理されます。エージェントの承認は必要ありません。以下の認証プラグインモジュールが提供されます。

- **ディレクトリーベースの登録**エンドエンティティーは、ユーザー ID とパスワード、またはその DN とパスワードを使用して LDAP ディレクトリーに対して認証されます。「ディレクトリーベースの認証の設定」を参照してください。
- **PIN ベースの登録。**エンドエンティティーは、ディレクトリーエントリーのユーザー ID、パスワード、および PIN セットを使用して LDAP ディレクトリーに対して認証されます。「PIN ベースの登録の設定」を参照してください。
- **証明書ベースの認証の使用。**ある種のエンティティー (エンドユーザーとサーバーやトークンなどの他のエンティティーの両方) は、CA によって発行された ID を証明する証明書を使用して CA に対して認証されます。これは、更新プロセスの認証に元の証明書が提示される、更新に最も一般的に使用されます。「証明書ベースの認証の使用」を参照してください。
- **AgentCertAuth.**このメソッドは、リクエストを送信したエンティティーがサブシステムエージェントとして認証される場合に証明書要求を自動的に承認します。ユーザーは、エージェント証明書を提示してエージェントとして認証します。提示された証明書がサブシステムでエージェント証明書として認識される場合、CA は証明書要求を自動的に処理します。

  この形式の自動認証は、サーバー証明書を登録する証明書プロファイルに関連付けることができます。

  このプラグインはデフォルトで有効になっており、パラメーターはありません。
- **フラットファイルベースの登録。**ルーター (SCEP) の登録専用に使用されるテキストファイルには、IP アドレス、ホスト名、またはその他の識別子のリストと、通常はランダムな PIN であるパスワードが含まれています。ルーターはその ID と PIN を使用して CA に対して認証し、CA は提示する認証情報をテキストファイルの ID の一覧と比較します。「フラットファイル認証の設定」を参照してください。

10.2.1. ディレクトリーベースの認証の設定

UidPwdDirAuth および **UdnPwdDirAuth** プラグインモジュールは、ディレクトリーベースの認証を実装します。LDAP ディレクトリーに対して認証するユーザー ID または DN およびパスワードを指定して、証明書にエンドユーザーを登録します。

1. **UidPwdDirAuth** または **UdnPwdDirAuth** 認証モジュールのいずれかのインスタンスを作成して、インスタンスを設定します。

   a. CA コンソールを開きます。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

   b. **Configuration** タブで、ナビゲーションツリーの **Authentication** を選択します。

   右側のペインには、現在設定されている認証インスタンスを一覧表示する **Authentication Instance** タブが表示されます。

   注記

   UidPwdDirAuth プラグインはデフォルトで有効です。

c. **Add** をクリックします。

Select Authentication plug-in Implementation ウインドウが表示されます。

d. ユーザー ID およびパスワード認証に **UidPwdDirAuth** を選択するか、DN およびパスワード認証には **UdnPwdDirAuth** を選択します。

e. **Authentication Instance Editor** ウィンドウで、以下のフィールドに入力します。

- **Authentication Instance ID**。デフォルトのインスタンス名を許可するか、新しい名前を入力します。
- **dnpattern。** ディレクトリー属性およびエントリー DN から形成するサブジェクト名パターンを表す文字列を指定します。
- **ldapStringAttributes.**エンドエンティティーの **認証** として考慮されるべき LDAP 文字列属性の一覧を指定します。これらの属性に対応する値は、認証ディレクトリーから認証トークンにコピーし、証明書プロファイルによりサブジェクト名を生成するために使用されます。このパラメーターの値の入力は任意です。
- **ldapByteAttributes.**エンドエンティティーの **認証** として考慮されるべき LDAP バイト (バイナリー) 属性の一覧を指定します。指定した場合、これらの属性に対応する値は、ユーザーの証明書への追加情報の追加など、他のモジュールで使用するために認証ディレクトリーから認証トークンにコピーされます。

  このパラメーターの値の入力は任意です。
- **ldap.ldapconn.host.**認証ディレクトリーの完全修飾 DNS ホスト名を指定します。
- **ldap.ldapconn.port.**認証ディレクトリーが要求をリッスンする TCP/IP ポートを指定します。**ldap.ldapconn.secureConn.** チェックボックスを選択した場合、これは SSL ポート番号になります。
- **ldap.ldapconn.secureConn.**認証ディレクトリーが Certificate System からの要求をリッスンするポートのタイプ (SSL または非 SSL) を指定します。これが SSL ポートである場合に選択します。
- **ldap.ldapconn.version.**LDAP プロトコルのバージョンの **2** または **3** を指定します。バージョン 3.x 以降のすべての Directory Server は LDAPv3 であるため、デフォルトは **3** です。
- **ldap.basedn.**認証ディレクトリーを検索するためにベース DN を指定します。サーバーは、HTTP 入力 (ユーザーが登録フォームに入るもの) とベース DN の **uid** フィールド値を使用して LDAP 検索フィルターを構築します。
- **ldap.minConns.**認証ディレクトリーで許可される最小接続数を指定します。許容値は、**1** から **3** です。
- **ldap.maxConns.**認証ディレクトリーで許可される接続の最大数を指定します。許容値は、**3** から **10** です。

f. **OK** をクリックします。認証インスタンスが設定され、有効になっている。

2. 特定の証明書のポリシーを設定して、ユーザーの登録に使用する証明書プロファイルを設定します。証明書プロファイルの入力を設定して登録フォームをカスタマイズします。また、ユーザーを認証するためにプラグインが必要とする情報の入力を含めます。デフォルトの入力に、

収集する必要のあるすべての情報が含まれていない場合には、サードパーティーツールで作成した要求を送信します。

プロファイルの設定に関する詳細は、「サブジェクト代替名への LDAP ディレクトリー属性値およびその他の情報の挿入」 を参照してください。

注記

pkiconsole が非推奨になりました。

バインドされた LDAP 接続の設定
一部の環境では、認証に使用される LDAP サーバーの匿名バインドを禁止する必要があります。CA と LDAP サーバーとの間にバインドされた接続を作成するには、以下の設定を変更する必要があります。

- **CS.cfg** の以下の例に従って、ディレクトリーベースの認証を設定します。

```
auths.instance.UserDirEnrollment.ldap.ldapBoundConn=true
auths.instance.UserDirEnrollment.ldap.ldapauth.authtype=BasicAuth
auths.instance.UserDirEnrollment.ldap.ldapauth.bindDN=cn=Directory Manager
auths.instance.UserDirEnrollment.ldap.ldapauth.bindPWPrompt=externalLDAP
externalLDAP.authPrefix=auths.instance.UserDirEnrollment
cms.passwordlist=internaldb,replicationdb,externalLDAP
```

  bindPWPrompt は、**password.conf** ファイルで使用されるタグまたはプロンプトです。また、**cms.passwordlist** オプションおよび **authPrefix** オプション で使用される名前でもあります。

- **CS.cfg** からタグまたはプロンプトを **password.conf** でパスワードとともに追加します。

```
externalLDAP=your_password
```

外部承認の設定
また、ディレクトリーベースの認証プラグインを設定して、ユーザーのグループメンバーシップを評価することもできます。このプラグインを設定するには、**CS.cfg** に以下のオプションを設定する必要があります。

- **groupsEnable** は、グループの取得を可能にするブール値オプションです。デフォルト値は **false** です。
- **groupsBasedn** はグループのベース DN です。これは、デフォルト **basedn** と異なる場合に指定する必要があります。
- **group** は、グループの DN コンポーネントです。デフォルト値は **ou=groups** です。
- **groupObjectClass** は、グループオブジェクトクラス **groupofuniquenames**、**groupofnames** のいずれかです。デフォルト値は **groupofuniquenames** です。
- **groupUseridName** は、グループオブジェクトメンバー属性のユーザー ID 属性の名前です。デフォルト値は **(cn=*)** です。
- **useridName** は、ユーザー ID DN コンポーネントの名前です。デフォルト値は **uid** です。
- **searchGroupUserByUserdn** は、**userdn** または **${groupUserIdName}=${uid}** 属性のグループオブジェクトメンバー属性を検索するかどうかを決定するブール値オプションです。デフォルト値は **true** です。

以下に例を示します。

```
auths.instance.UserDirEnrollment.pluginName=UidPwdDirAuth
auths.instance.UserDirEnrollment.ldap.basedn=cn=users,cn=accounts,dc=local
auths.instance.UserDirEnrollment.ldap.groupObjectClass=groupofnames
auths.instance.UserDirEnrollment.ldap.groups=cn=groups
auths.instance.UserDirEnrollment.ldap.groupsBasedn=cn=accounts,dc=local
auths.instance.UserDirEnrollment.ldap.groupsEnable=true
auths.instance.UserDirEnrollment.ldap.ldapconn.host=local
auths.instance.UserDirEnrollment.ldap.ldapconn.port=636
auths.instance.UserDirEnrollment.ldap.ldapconn.secureConn=true
```

最後に、/***instance_path***/**ca**/**profiles**/**ca**/***profile_id***.**cfg** ファイルを変更して、**CS.cfg** に定義された **UserDirEnrollment** 認証インスタンスを使用するようにプロファイルを設定し、および必要に応じて、グループに基づく許可用の ACL を提供します。以下に例を示します。

```
auth.instance_id=UserDirEnrollment
auths.acl=group="cn=devlab-access,ou=engineering,dc=example,dc=com"
```

10.2.2. PIN ベースの登録の設定

PIN ベースの認証では、LDAP ディレクトリーでユーザーごとに PIN を設定し、それらの PIN をユーザーに配布してから、証明書要求に入力するときにユーザーにユーザー ID とパスワードとともに PIN を提供してもらいます。その後、ユーザーはユーザー ID とパスワードを使用して LDAP ディレクトリーと LDAP エントリーの PIN に対して認証されます。ユーザーが認証に成功すると、リクエストは自動的に処理され、新しい証明書が発行されます。

Certificate System は、Directory Server に必要なスキーマを Directory Server に追加し、各ユーザーの PIN を生成するツール **setpin** を提供します。

PIN ツールは、以下の機能を実行します。

- PIN に必要なスキーマを LDAP ディレクトリーに追加します。
- 設定した PIN に読み取り/書き込みパーミッションを持つ PIN マネージャーユーザーを追加します。
- PIN の使用後に PIN の削除を許可するように ACI を設定し、PIN マネージャーに PIN の読み取り/書き込み権限を付与し、PIN を作成または変更できないようにします。
- 各ユーザーエントリーに PIN を作成します。

注記

このツールは、『Certificate System Command-Line Tools Guide』 に記載されています。

1. PIN ツールを使用して PIN に必要なスキーマを追加し、ユーザーエントリーに PIN を追加してから PIN をユーザーに配布します。
   a. /**usr**/**share**/**pki**/**native-tools**/ ディレクトリーを開きます。
   b. テキストエディターで **setpin.conf** ファイルを開きます。
   c. ファイルに概説されている手順に従って、適切な変更を加えます。

通常、更新が必要なパラメーターは、Directory Server のホスト名、Directory Manager のバインドパスワード、および PIN マネージャーのパスワードです。

d. **setpin.conf** ファイルをポイントする **optfile** オプションを指定して、**setpin** コマンドを実行します。

```
setpin optfile=/usr/share/pki/native-tools/setpin.conf
```

このツールは、新しい属性 (デフォルトでは **pin**) および新しいオブジェクトクラス (デフォルトは **pinPerson**) でスキーマを変更し、**pinmanager** ユーザーを作成し、ACI を設定して、**pinmanager** ユーザーのみが **pin** 属性を編集できるようにします。

e. 特定のユーザーエントリーの PIN を生成するか、ユーザー定義の PIN を指定する場合は、これらのエントリーの DN を指定して入力ファイルを作成します。たとえば、以下のようになります。

```
dn:uid=bjensen,ou=people,dc=example,dc=com
dn:uid=jsmith,ou=people,dc=example,dc=com
dn:jtyler,ou=people,dc=example,dc=com
...
```

入力ファイルを構築する方法は、『Certificate System コマンドラインツールガイド』の PIN ジェネレーターの章を参照してください。

f. **setpin** コマンドのセットアップモードを無効にします。**setup** 行をコメントアウトするか、値を no に変更します。

```
vim /usr/share/pki/native-tools/setpin.conf

setup=no
```

セットアップモードでは、必要なユーザーとオブジェクトクラスが作成されますが、セットアップモードでは、ツールは PIN を生成しません。

g. **setpin** コマンドを実行して、ディレクトリーに PIN を作成します。

ヒント

実際にディレクトリーを実際には変更せずに PIN のリストを生成する **write** オプションを使用せずに、最初にツールをテスト実行します。

以下に例を示します。

```
setpin host=yourhost port=9446 length=11 input=infile output=outfile write
"binddn=cn=pinmanager,o=example.com" bindpw="password" basedn=o=example.com
"filter=(uid=u*)" hash=sha256
```

警告

hash 引数を **none** に設定しないでください。**hash=none** を付けて **setpin** コマンドを実行すると、ピンはプレーンテキストとしてユーザー LDAP エントリーに保存されます。

h. 必要な認証方法の設定が完了したら、出力ファイルを使用して PIN をユーザーに配信します。

PIN ベースの登録が機能することを確認したら、PIN をユーザーに配信して、登録時に使用できるようにします。PIN のプライバシーを保護するには、安全で帯域外での配信方法を使用します。

2. 証明書プロファイルにポリシーを設定して、ユーザーを登録します。証明書プロファイルポリシーの詳細は、*3章証明書を発行するルール (証明書プロファイル) の作成* を参照してください。

3. **UidPwdPinDirAuth** 認証プラグインのインスタンスを作成して設定します。

a. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

b. **Configuration** タブで、ナビゲーションツリーの **Authentication** を選択します。

右側のペインには、現在設定されている認証インスタンスを一覧表示する **Authentication Instance** タブが表示されます。

c. **Add** をクリックします。

Select Authentication plug-in Implementation ウインドウが表示されます。

d. **UidPwdPinDirAuth** プラグインモジュールを選択します。

e. **Authentication Instance Editor** ウィンドウで、以下のフィールドに入力します。

- **Authentication Instance ID**。デフォルトのインスタンス名を使用するか、新しい名前を入力します。
- **removePin.**エンドユーザーの認証に成功した後に、認証ディレクトリーから PIN を削除するかどうかを設定します。ディレクトリーから PIN を削除すると、ユーザーが複数回登録できなくなるため、複数の証明書を取得できなくなります。
- **pinAttr.**PIN の認証ディレクトリー属性を指定します。**PIN Generator** ユーティリティーは、**setpin.conf** ファイルの **objectclass** パラメーターの値に属性を設定します。このパラメーターのデフォルト値は **pin** です。
- **dnpattern。** ディレクトリー属性およびエントリー DN から形成するサブジェクト名パターンを表す文字列を指定します。
- **ldapStringAttributes.**エンドエンティティーの **認証** として考慮されるべき LDAP 文字列属性の一覧を指定します。このパラメーターの値の入力は任意です。

- **ldapByteAttributes.**エンドエンティティーの **認証** として考慮されるべき LDAP バイト (バイナリー) 属性の一覧を指定します。指定した場合、これらの属性に対応する値は、ユーザーの証明書への追加情報の追加など、他のモジュールで使用するために認証ディレクトリーから認証トークンにコピーされます。

  このパラメーターの値の入力は任意です。

- **ldap.ldapconn.host.**認証ディレクトリーの完全修飾 DNS ホスト名を指定します。

- **ldap.ldapconn.port.**認証ディレクトリーが Certificate System からのリクエストをリッスンする TCP/IP ポートを指定します。

- **ldap.ldapconn.secureConn.**認証ディレクトリーが要求をリッスンするポートの、SSL タイプ、SSL、または非 SSL を指定します。これが SSL ポートである場合に選択します。

- **ldap.ldapconn.version.**LDAP プロトコルのバージョンの **2** または **3** を指定します。デフォルトでは、3.x 以降のすべての Directory Server バージョンが LDAPv3 であるため、これは **3** になります。

- **ldap.ldapAuthentication.bindDN.**認証ディレクトリーから PIN を削除する際にバインドするユーザーエントリーを指定します。**removePin** チェックボックスが選択されている場合に限り、このパラメーターを指定します。ディレクトリー内の PIN 属性のみを変更するパーミッションを持つ別のユーザーエントリーを作成して使用することが推奨されます。たとえば、ディレクトリーのコンテンツ全体を変更する権限があるため、Directory Manager のエントリーを使用しないでください。

- **password.ldap.ldapauthbindDN** パラメーターで指定された DN に関連付けられたパスワードを指定します。変更を保存したら、サーバーはパスワードをシングルサインオンパスワードキャッシュに保存して、後続の起動時に使用します。このパラメーターは、**removePin** チェックボックスが選択されている場合にのみ設定する必要があります。

- **ldap.ldapAuthentication.clientCertNickname.**PIN を削除する認証ディレクトリーへの SSL クライアント認証に使用する証明書のニックネームを指定します。証明書が有効で、認証ディレクトリーの証明書データベースで信頼できる CA によって署名されていることを確認し、認証ディレクトリーの **certmap.conf** ファイルがディレクトリーの DN に正しくマッピングするように設定されていることを確認してください。これは PIN の削除のみに必要です。

- **ldap.ldapAuthentication.authtype.**認証ディレクトリーから PIN を削除するのに必要な認証タイプ、Basic 認証、または SSL クライアント認証を指定します。

  - **BasicAuth** は Basic 認証を指定します。このオプションを使用すると、**ldap.ldapAuthentication.bindDN** および **password** パラメーターの正しい値を入力します。サーバーは **ldap.ldapAuthentication.bindDN** 属性の DN を使用してディレクトリーにバインドします。

  - **SslClientAuth** は、SSL クライアント認証を指定します。このオプションを使用すると、**ldap.ldapconn.secureConn** パラメーター の値を **true** に設定し、証明書のニックネームの **ldap.ldapAuthentication.clientCertNickname** パラメーターの値を、SSL クライアント認証に使用する証明書のニックネームに設定します。

- **ldap.basedn.**認証ディレクトリーを検索するためのベース DN を指定します。サーバーは、HTTP インプット (ユーザーが登録フォームに入力するもの) および LDAP 検索フィルターを構築するためのベース DN から **uid** フィールドの値を使用します。

- **ldap.minConns.**認証ディレクトリーで許可される最小接続数を指定します。許容値は、**1** から **3** です。
- **ldap.maxConns.**認証ディレクトリーで許可される接続の最大数を指定します。許容値は、**3** から **10** です。

f. **OK** をクリックします。

4. 証明書プロファイルで入力を設定して、登録フォームをカスタマイズします。ユーザーの認証にプラグインが必要とする情報を含めます。デフォルトの入力に、収集する必要のあるすべての情報が含まれていない場合には、サードパーティーツールで作成した要求を送信します。

注記

pkiconsole が非推奨になりました。

10.2.3. 証明書ベースの認証の使用

証明書ベースの認証 は、要求側の ID を検証し、送信されるリクエストを自動的に検証し、認証する証明書が表示される場合です。これは、元の証明書がユーザー、サーバー、およびアプリケーションによって提示され、その証明書が要求を認証するのに使用される場合に、更新プロセスに最も一般的に使用されます。

証明書の初回要求に証明書ベースの認証を使用する場合は、その他の状況があります。たとえば、トークンに汎用証明書を一括で読み込んで、ユーザーがユーザー証明書に登録するときにユーザーを認証するために使用することも、ユーザーに署名証明書を発行して、暗号化証明書の要求を認証するために使用することもできます。

証明書ベースの認証モジュール **SSLclientCertAuth** はデフォルトで有効になっており、この認証方法は任意のカスタム証明書プロファイルで参照できます。

10.2.4. フラットファイル認証の設定

ルーター証明書は無作為に生成される PIN を使用して登録され、認証されます。CA は **flatFileAuth** 認証モジュールを使用して、ルーターの認証情報が含まれるテキストファイルを処理します。

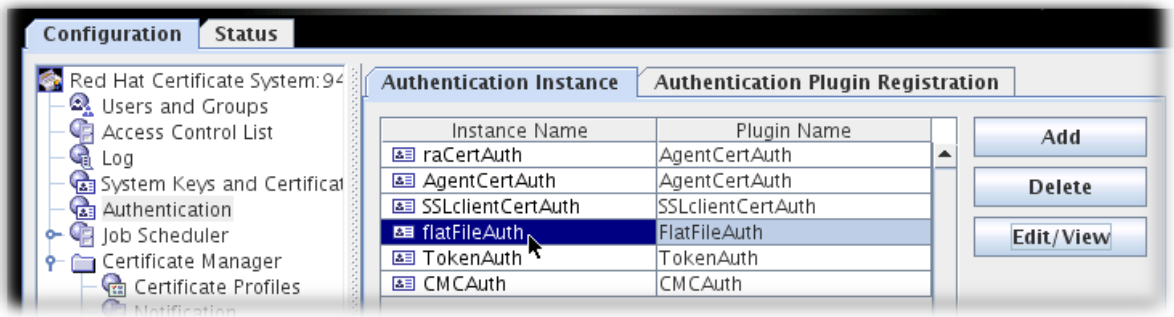
10.2.4.1. flatFileAuth モジュールの設定

フラットファイル認証はすでに SCEP 登録用に設定されていますが、フラットファイルの場所とその認証パラメーターを編集できます。

1. CA コンソールを開きます。

```
pkiconsole https://server.example.com:8443/ca
```

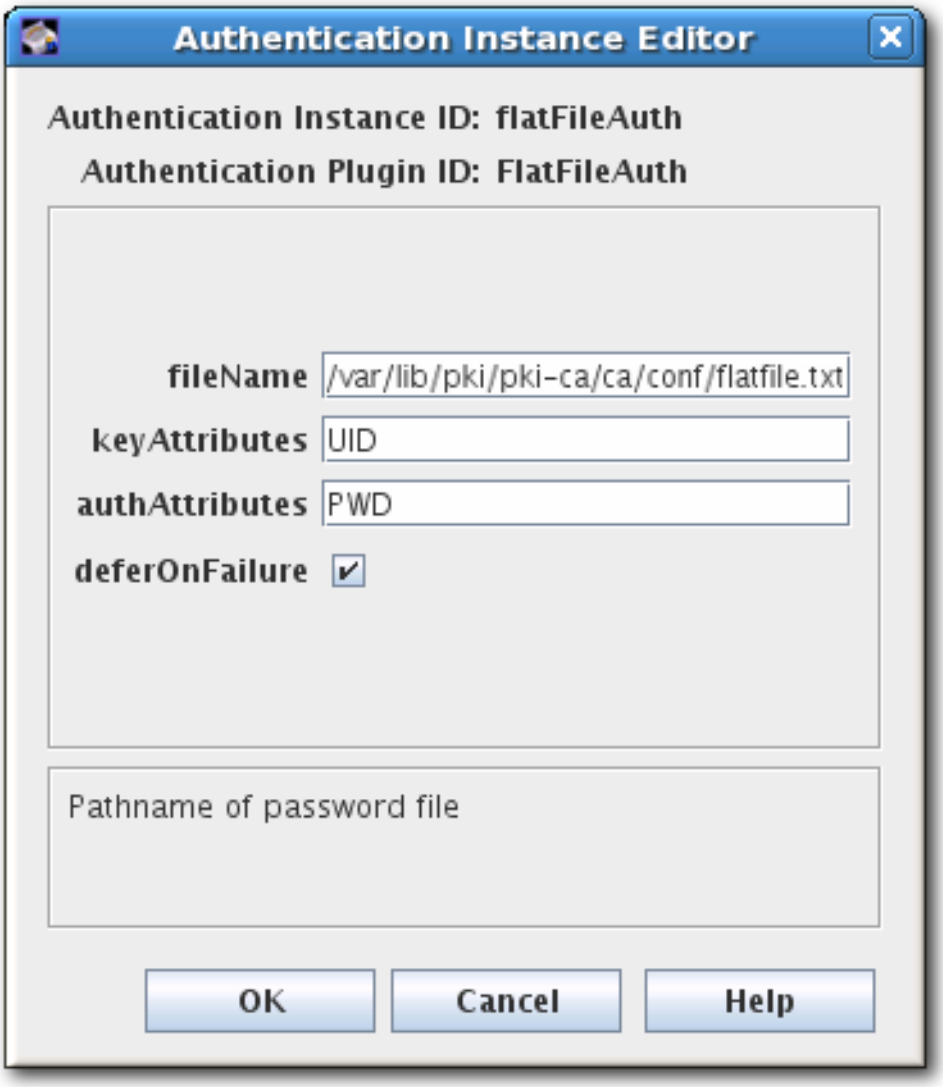
2. **Configuration** タブで、ナビゲーションツリーの **Authentication** を選択します。
3. **flatFileAuth** 認証モジュールを選択します。



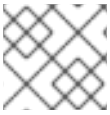
4. **Edit/View** をクリックします。

5. ファイルの場所と名前を変更するには、**fileName** フィールドをリセットします。

   authentication name パラメーターを変更するには、**keyAttributes** の値を、CN などの SCEP 登録フォームで送信された別の値にリセットします。また、**UID,CN** のようにコンマで区切って複数の name パラメーターを使用することもできます。パスワードパラメーター名を変更するには、**authAttributes** フィールドをリセットします。



6. 編集を保存します。



注記

pkiconsole が非推奨になりました。

10.2.4.2. flatfile.txt の編集

同じ **flatfile.txt** ファイルを使用して、SCEP 登録をすべて認証します。このファイルは、新規 PIN がルーターに発行されるたびに手動で更新する必要があります。

デフォルトでは、このファイルは **/var/lib/pki/pki-ca/ca/conf/** にあり、認証エントリーごとに 2 つのパラメーター、サイトの UID (通常は IPv4 または IPv6)、およびルーターが発行する PIN の 2 つのパラメーターを指定します。

```
UID:192.168.123.123
PIN:HU89dj
```

各エントリーの後には空白行が続く必要があります。以下に例を示します。

```
UID:192.168.123.123
PIN:HU89dj

UID:12.255.80.13
PIN:fiowIO89

UID:0.100.0.100
PIN:GRIOjisf
```

認証エントリーが空の行で区切られていない場合、ルーターが CA に対して認証を試みたときに、失敗します。以下に例を示します。

```
... flatfile.txt entry ...
UID:192.168.123.123
PIN:HU89dj
UID:12.255.80.13
PIN:fiowIO89

... error log entry ...
[13/Jun/2020:13:03:09][http-9180-Processor24]: FlatFileAuth: authenticating user: finding user from key: 192.168.123.123
[13/Jun/2020:13:03:09][http-9180-Processor24]: FlatFileAuth: User not found in password file.
```

10.3. CMC 認証プラグイン

CMC 登録により、登録クライアントは認証に CMC 認証プラグインを使用できます。これにより、証明書要求は、プラグインに応じて、エージェント証明書またはユーザー証明書のいずれかで事前署名されます。Certificate Manager は、有効な証明書で署名された要求を受信すると、証明書を自動的に発行します。

CMC 認証プラグインは、クライアントに CMC 失効も提供します。CMC の失効により、クライアントは、エージェント証明書または証明書を所有する検証可能なユーザーによって署名された証明書要求を取得し、そのような要求を Certificate Manager に送信できます。Certificate Manager は、有効な証明書で署名された要求を受け取ると、証明書を自動的に取り消します。

Certificate System は、次の CMC 認証プラグインを提供します。

CMCAuth

CA エージェントが CMC 要求に署名する場合は、このプラグインを使用します。

CMCAuth プラグインを使用するには、登録プロファイルで以下を設定します。

```
auth.instance_id=CMCAuth
```

デフォルトでは、以下の登録プロファイルは **CMCAuth** プラグインを使用します。

- システム証明書の場合:
  - **caCMCauditSigningCert**
  - **caCMCcaCert**
  - **caCMCECserverCert**
  - **caCMCECsubsystemCert**
  - **caCMCECUserCert**
  - **caCMCkraStorageCert**
  - **caCMCkraTransportCert**
  - **caCMCocspCert**
  - **caCMCserverCert**
  - **caCMCsubsystemCert**
- ユーザー証明書の場合:
  - **caCMCUserCert**
  - **caECFullCMCUserCert**
  - **caFullCMCUserCert**

CMCUserSignedAuth

署名付きまたは SharedSecret ベースの CMC 要求を送信する場合は、このプラグインを使用します。

CMCUserSignedAuth プラグインを使用するには、登録プロファイルに以下を設定します。

```
auth.instance_id=CMCUserSignedAuth
```

ユーザー署名の CMC 要求は、要求された証明書と同じ **subjectDN** 属性が含まれるユーザーの証明書で署名する必要があります。ユーザーが署名した CMC 要求を使用できるのは、ユーザーが他の証明書のユーザー ID を証明するために使用できる署名証明書を既に取得している場合のみです。

SharedSecret ベースの CMC 要求は、要求が要求自体の秘密鍵によって署名されたことを意味します。この場合、CMC 要求は認証に共有秘密メカニズムを使用する必要があります。SharedSecret ベースの CMC 要求は通常、ユーザーの最初の署名証明書を取得するために使用され、後で他の証明書を取得するために使用されます。詳細は、「CMC SharedSecret 認証」を参照してください。

デフォルトでは、以下の登録プロファイルは、**CMCUserSignedAuth** プラグインを使用します。

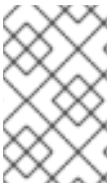
- **caFullCMCUserSignedCert**
- **caECFullCMCUserSignedCert**
- **caFullCMCSharedTokenCert**
- **caECFullCMCSharedTokenCert**

10.4. CMC SHAREDSECRET 認証

Shared Secret 機能を使用して、ユーザーがサーバーに署名されていないリクエストを送信できるようにします。たとえば、ユーザーが最初の署名証明書を取得する場合は、これが必要になります。この署名証明書は、後でこのユーザーの他の証明書に署名するために使用できます。

10.4.1. 共有シークレットトークンの作成

詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『共有シークレットのワークフロー』セクションを参照してください。状態に応じて、エンドエンティティーユーザーまたは管理者が共有シークレットトークンを作成します。



注記

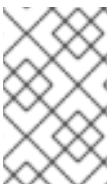
共有シークレットトークンを使用するには、Certificate System で RSA 発行の保護証明書を使用する必要があります。詳細は、RHCS P 計画、インストール、およびデプロイメントのガイドの共有シークレット機能の有効化セクションを参照してください。

Shared Secret Token を作成するには、以下を入力します。

```
# CMCSharedToken -d /home/user_name/.dogtag/ -p NSS_password \
    -s "CMC_enrollment_password" -o /home/user_name/CMC_shared_token.b64 \
    -n "issuance_protection_certificate_nickname"
```

HSM を使用する場合は、さらに HSM セキュリティートークン名を設定するコマンドの **-h *token_name*** オプションを渡します。

CMCSharedToken ユーティリティーの詳細は、CMCSharedToken(8) の man ページを参照してください。



注記

生成されたトークンは暗号化され、パスワードを認識したユーザーのみになります。CA 管理者がユーザーのトークンを生成する場合、管理者はセキュアな方法でユーザーにパスワードを提供する必要があります。

Shared Token を作成したら、管理者はトークンをユーザーまたは証明書レコードに追加する必要があります。詳細は、「CMC 共有シークレットの設定」 を参照してください。

10.4.2. CMC 共有シークレットの設定

管理者は、計画されるアクションに応じて、ユーザーまたは証明書の LDAP エントリーに生成した後に Shared Secret Token を保存する必要があります。

ワークフローおよび Shared Secret を使用する場合の詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『共有シークレットのワークフロー』セクションを参照してください。

10.4.2.1. 証明書の登録用ユーザーエントリーへの CMC 共有シークレットの追加

証明書の登録に Shared Secret Token を使用するには、ユーザーの LDAP エントリーに管理者として保存します。

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: uid=user_name,ou=People,dc=example,dc=com
changetype: modify
replace: shrTok
shrTok: base64-encoded_token
```

10.4.2.2. 証明書失効用の証明書への CMC 共有シークレットの追加

証明書失効に Shared Secret Token を使用するには、取り消される証明書の LDAP エントリーに管理者として保存します。

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=certificate_id,ou=certificateRepository,ou=ca,o=pki-tomcat-CA
changetype: modify
replace: shrTok
shrTok: base64-encoded_token
```

10.5. 登録のテスト

プロファイルを使用した登録のテストの詳細は、*3章証明書を発行するルール (証明書プロファイル) の作成* を参照してください。認証方法セットを使用して、エンドユーザーが正常に証明書を登録できるかどうかをテストするには、以下を実行します。

1. エンドエンティティーを開きます。

   ```
   https://server.example.com:8443/ca/ee/ca
   ```

2. **登録** タブで、カスタマイズされた登録フォームを開きます。
3. 値を入力してリクエストを送信します。
4. プロンプトが表示されたら、キーデータベースにパスワードを入力します。
5. 正しいパスワードを入力すると、クライアントはキーペアを生成します。

   キー生成のプロセスを中断しないでください。キーの生成が完了すると、リクエストはサーバーに送信され、証明書を発行します。サーバーは、証明書プロファイルへの要求を許可し、要求がすべての要件を満たす場合にのみ証明書を発行します。

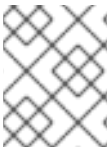
   証明書を発行したら、ブラウザーに証明書をインストールします。

6. 証明書がブラウザーの証明書データベースにインストールされていることを確認します。

7. PIN ベースのディレクトリー認証が PIN の削除で設定されている場合は、同じ PIN を使用して別の証明書を再登録します。リクエストを拒否する必要があります。

10.6. カスタム認証プラグインの登録

カスタム認証プラグインモジュールは、CA コンソールから登録できます。認証プラグインモジュールは、CA コンソールからも削除できます。モジュールを削除する前に、そのモジュールに基づいたインスタンスを削除します。



注記

カスタムプラグインを記述する場合は、認証プラグインのチュートリアル を参照してください。

1. カスタム認証クラスを作成します。この例では、カスタム認証プラグインは、**UidPwdDirAuthenticationTestms.java** となります。

2. 新しいクラスをコンパイルします。

```
javac -d . -classpath $CLASSPATH UidPwdDirAuthenticationTestms.java
```

3. CA が登録フォームからカスタムクラスにアクセスできるように、CA の **WEB-INF** Web ディレクトリーにディレクトリーを作成します。

```
mkdir /usr/share/pki/ca/webapps/ca/WEB-INF/classes
```

4. 新しいプラグインファイルを新しい **class** ディレクトリーにコピーし、所有者を Certificate System system user (**pkiuser**) に設定します。

```
cp -pr com /usr/share/pki/ca/webapps/ca/WEB-INF/classes

chown -R pkiuser:pkiuser /usr/share/pki/ca/webapps/ca/WEB-INF/classes
```

5. コンソールにログインします。

```
pkiconsole https://server.example.com:8443/ca
```

6. プラグインを登録します。

   a. **Configuration** タブで、ナビゲーションツリーの **Authentication** をクリックします。

   b. 右側のペインで、**Authentication Plug-in Registration** をクリックします。

      タブは、登録済みのモジュールの一覧を表示します。

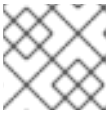
   c. プラグインを登録するには、**Register** をクリックします。

      Register Authentication Plug-in Implementation 画面が表示されます。

   d. 2 つのフィールドを入力して登録するモジュールを指定します。

- **プラグイン名。**モジュールの名前。
- **クラス名。**このモジュールのクラスのフルネーム。これは、実装 Java™ クラスへのパスです。このクラスがパッケージに含まれる場合は、パッケージ名を含めます。たとえば、**com.customplugins** という名前のパッケージに **customAuth** という名前のクラスを登録するには、クラス名は **com.customplugins.customAuth** になります。

7. モジュールを登録したら、モジュールをアクティブな認証インスタンスとして追加します。
   a. **Configuration** タブで、ナビゲーションツリーの **Authentication** をクリックします。
   b. 右側のペインで、**Authentication Instance** タブをクリックします。
   c. **Add** をクリックします。
   d. 一覧からカスタムモジュール **UidPwdDirAuthenticationTestms.java** を選択し、リストからモジュールを追加します。モジュールに適した設定を入力します。



注記

pkiconsole が非推奨になりました。

8. 新しい認証モジュールを使用するために、新しいエンドエンティティーの登録フォームを作成します。

```
cd /var/lib/pki/pki-tomcat/ca/profiles/ca

cp -p caDirUserCert.cfg caDirUserCertTestms.cfg

vi caDirUserCertTestms.cfg

desc=Test ms - This certificate profile is for enrolling user certificates with directory-based
authentication.
visible=true
enable=true
enableBy=admin
name=Test ms - Directory-Authenticated User Dual-Use Certificate Enrollment
auth.instance_id=testms
...
```

9. 新規プロファイルを CA の **CS.cfg** ファイルに追加します。

ヒント

CS.cfg ファイルを編集する前にバックアップします。

```
vim /var/lib/pki/instance-name/ca/conf/CS.cfg

profile.list=caUserCert,caDualCert,caSignedLogCert,caTPSCert,caRARouterCert,caRouterCer
t,caServerCert,caOtherCert,caCACert,caInstallCACert,caRACert,caOCSPCert,caTransportCe
rt,caDirUserCert,caAgentServerCert,caAgentFileSigning,caCMCUserCert,caFullCMCUserCert
,caSimpleCMCUserCert,caTokenDeviceKeyEnrollment,caTokenUserEncryptionKeyEnrollment,
caTokenUserSigningKeyEnrollment,caTempTokenDeviceKeyEnrollment,caTempTokenUserEn
cryptionKeyEnrollment,caTempTokenUserSigningKeyEnrollment,caAdminCert,caInternalAuthS
```

```
erverCert,caInternalAuthTransportCert,caInternalAuthKRAstorageCert,caInternalAuthSubsyste
mCert,caInternalAuthOCSPCert,DomainController,caDirUserCertTestms
...
profile.caDirUserCertTestms.class_id=caEnrollImpl
profile.caDirUserCertTestms.config=/var/lib/pki/pki-
tomcat/ca/profiles/ca/caDirUserCertTestms.cfg
```

10. CA を再起動します。

```
pki-server restart instance_name
```

10.7. コマンドラインを使用した証明書ステータスの手動確認

証明書要求を確認するには、証明書要求を承認する適切なパーミッションを持つエージェントとして認証されていることを確認してください。**pki** コマンドラインインターフェイスの設定に関する詳細は、「pki CLI の初期化」 を参照してください。

要求を確認するには、以下を実行します。

1. 保留中の証明書要求の一覧を表示します。

```
$ pki agent_authentication_parameters ca-cert-request-find --status pending
```

このコマンドは、保留中の証明書要求をすべて表示します。

2. 特定の証明書要求をダウンロードします。

```
$ pki agent_authentication_parameters ca-cert-request-review id --file request.xml
```

3. エディターまたは別のターミナルでエディターまたは別のターミナルで ***request.xml*** ファイルを開いて要求の内容を確認して、それが正当であることを確認します。その後、プロンプトに回答します。リクエストが有効な場合は、**approve** と回答して、**Enter** を押します。リクエストが無効の場合は **reject** と回答し、**Enter** を押します。組織は、セマンティックの相違点をサブスクライブして **reject** および **cancel** にすることができます。どちらも証明書は発行されません。

10.8. WEB インターフェイスを使用した証明書ステータスの手動による確認

1. Web ブラウザーで、以下の URL を開きます。

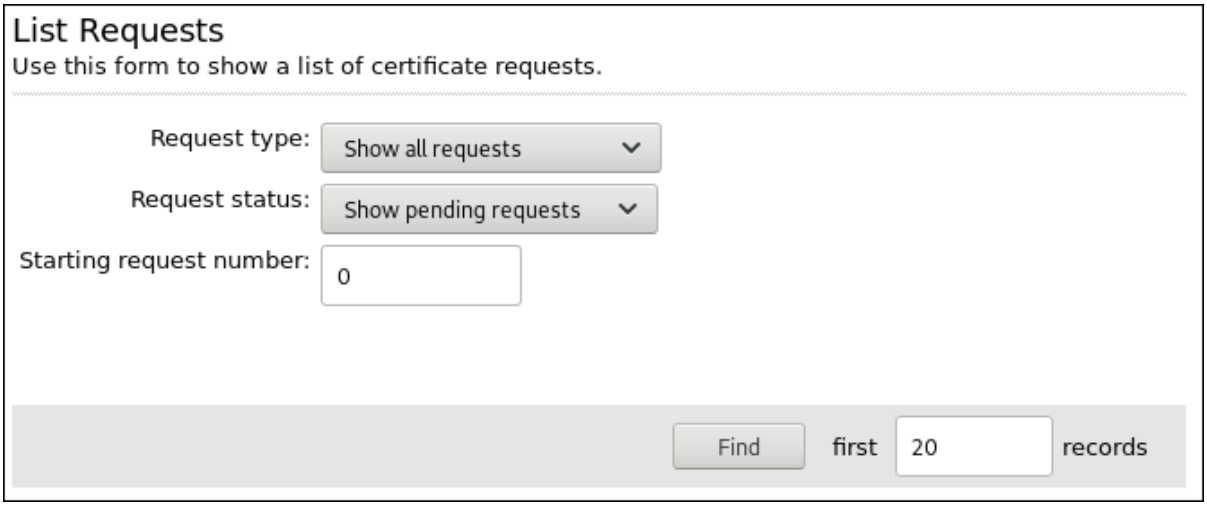
```
https://server_host_name:8443/ca/agent/ca
```

2. エージェントとして認証します。ユーザーとして認証し、ブラウザーの設定に関する詳細は、「ブラウザーの初期化」 を参照してください。

3. 左側のサイドバーの **List requests** リンクをクリックします。

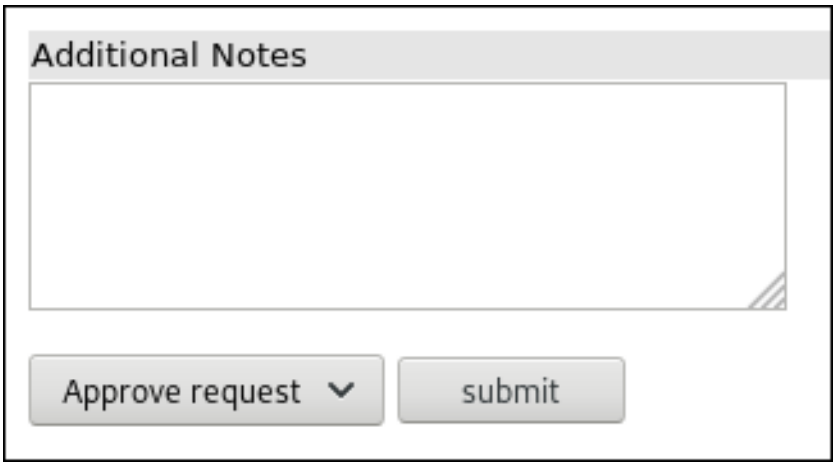
4. **Request type** の **Show all requests** を選択して、**Request status** の **Pending requests** を選択して要求をフィルタリングします。

5. 右下隅にある **Find** をクリックします。



6. 結果ページでは、確認を待機中の保留中のリクエストをすべて表示します。要求番号をクリックして、リクエストを確認します。

7. 要求情報を確認し、それが正当な要求であることを確認します。必要に応じて、ポリシー情報を変更して間違いを修正したり、証明書に必要な変更 (**not valid after** フィールドなど) を行ったりします。必要に応じて、追加の注記を残しておきます。



ドロップダウンメニューには、複数のレビューステータスの更新が含まれます。**Approve request** を選択してリクエストを承認するか、または **Reject request** を選択して否定してから、**Submit** をクリックします。組織は **Reject request** と **Cancel Request** とのセマンティックの相違点をサブスクライブできます。いずれの場合でも、証明書は発行されません。

第11章 証明書の登録の認可 (アクセス評価者)

本章では、アクセスエバリュエーターを使用した承認メカニズムを説明します。

11.1. 承認メカニズム

認証 メカニズムの他に、各登録プロファイルに独自の **承認** メカニズムがあるように設定できます。承認メカニズムは、認証が成功しないと実行されません。

承認メカニズムは、Access Evaluator プラグインフレームワークによって提供されます。アクセスエバリュエーターは、アクセス制御命令 (ACI) エントリーの評価に使用されるプラグ可能なクラスです。このメカニズムは、事前に定義された引数のリスト (つまり **type**、**op**、**value**) などを取り、**group='Certificate Manager Agents'** などの評価を評価し、評価の結果に応じてブール値を返す **評価** 方法を提供します。

11.2. デフォルトの評価者

Red Hat Certificate System は、デフォルトのエバリュエーターを 4 つ提供します。**CS.cfg** ファイルに、以下のエントリーがデフォルトで一覧表示されます。

```
accessEvaluator.impl.group.class=com.netscape.cms.evaluators.GroupAccessEvaluator
accessEvaluator.impl.ipaddress.class=com.netscape.cms.evaluators.IPAddressAccessEvaluator
accessEvaluator.impl.user.class=com.netscape.cms.evaluators.UserAccessEvaluator
accessEvaluator.impl.user_origreq.class=com.netscape.cms.evaluators.UserOrigReqAccessEvaluator
```

group アクセスエバリュエーターは、ユーザーのグループメンバーシッププロパティーを評価します。たとえば、以下の登録プロファイルエントリーでは、CA エージェントのみがそのプロファイルで登録できます。

```
authz.acl=group="Certificate Manager Agents"
```

ipaddress アクセスエバリュエーターは、要求側の IP アドレスを評価します。たとえば、以下の登録プロファイルエントリーでは、指定した IP アドレスを持つホストのみがそのプロファイルで登録を行えます。

```
authz.acl=ipaddress="a.b.c.d.e.f"
```

user アクセスエバリュエーターは、完全一致についてユーザー ID を評価します。たとえば、以下の登録プロファイルエントリーでは、リストされたユーザーと一致するユーザーのみが、そのプロファイルを使用した登録を行うことができます。

```
authz.acl=user="bob"
```

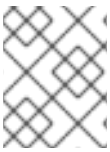
user_origreq アクセスエバリュエーターは、認証されたユーザーを、以前に一致した同等の要求に対して評価します。この特別なエバリュエーターは、更新を要求するユーザーが元の要求を所有するユーザーと同じであることを確認するために、更新を目的として特別に設計されています。たとえば、次の更新登録プロファイルエントリーでは、認証されたユーザーの UID は、更新を要求しているユーザーの UID と一致する必要があります。

```
authz.acl=user_origreq="auth_token.uid"
```

新しいエバリュエーターは現在のフレームワークで記述でき、CS コンソールから登録できます。デフォルトのエバリュエーターはテンプレートとして使用して、より多くのターゲットプラグインを拡張し、カスタマイズできます。

第12章 自動通知の使用

Certificate System は、証明書が発行または取り消されたときにエンドユーザーに、または新しい要求がエージェント要求キューに到着したときにエージェントに自動電子メール通知を送信するように設定できます。本章では、自動通知について説明し、送信される通知電子メールメッセージを有効化、設定、およびカスタマイズする方法を詳しく説明します。



注記

送信可能な通知の種類により、Certificate Manager のみが通知用に設定できます。このオプションは、他のサブシステムでは使用できません。

12.1. CA の自動通知について

自動通知は、指定されたイベント発生時に送信される電子メールメッセージです。システムは、システムを監視するリスナーを使用して、特定のイベントがいつ発生したかを判別します。イベントが発生すると、システムがトリガーされ、設定された受信者に電子メールが送信されます。各タイプの通知は、プレーンテキストまたは HTML のテンプレートを使用して、通知メッセージを作成します。テンプレートには、特定のイベントの正しい情報を入力するために展開されるテキストとトークンが含まれています。メッセージは、テンプレートに含まれるテキストおよびトークンを変更することでカスタマイズできます。HTML テンプレートは、さまざまな外観やフォーマット用にカスタマイズすることもできます。

12.1.1. 自動通知の種類

自動通知には、以下の 3 つのタイプがあります。

- **発行された証明書**。

  通知メッセージは、証明書を発行したユーザーに自動的に送信されます。ユーザーの証明書要求が拒否されると、拒否メッセージはユーザーに送信されます。

- **証明書失効**。

  ユーザー証明書が取り消されると、通知メッセージはユーザーに自動的に送信されます。

- **キューの要求**。

  エージェントに設定されたメールアドレスを使用して、要求がエージェント要求キューに入ると、通知メッセージが 1 つ以上のエージェントに自動的に送信されます。この通知タイプは、メッセージがキューに入るたびにメールを送信します。キュー内のジョブ要求の詳細は、「requestInQueueNotifier (RequestInQueueJob)」 を参照してください。

  キューのステータスに関する通知をエージェントに送信するジョブもあります。これには、特定の間隔でのキューステータスの概要が含まれます。

12.1.2. エンドエンティティーのメールアドレスの判断

通知システムは、最初に証明書要求または失効要求、次に証明書のサブジェクト名、最後に証明書の Subject Alternative Name 拡張子 (証明書にこの拡張子が含まれている場合) をチェックすることにより、エンドエンティティーの電子メールアドレスを決定します。電子メールアドレスが見つからない場合、通知は **Notification** パネルの **Sender's Email Address** フィールドで指定された電子メールアドレスに送信されます。

12.2. CA の自動通知の設定

12.2.1. コンソールで自動通知の設定

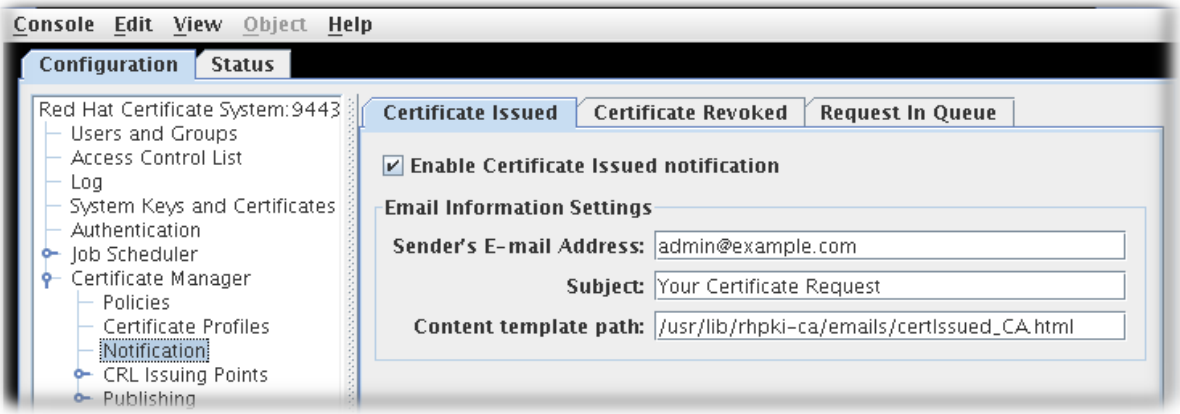
1. Certificate Manager Console を開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブを開きます。

3. 左側のナビゲーションツリーで **Certificate Manager** の見出しを開きます。次に **Notification** を選択します。

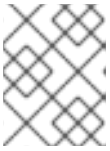
   Notification タブがウィンドウの右側に表示されます。



4. 通知は、新しく発行された証明書、取り消された証明書、および新しい証明書要求の 3 種類のイベントに対して送信できます。イベントの通知を送信するには、タブを選択し、**Enable** チェックボックスをオンにして、次のフィールドに情報を指定します。

   - **送信者のメールアドレス**。配信の問題が通知されるユーザーの完全なメールアドレスを入力します。
   - **Recipient's E-mail Address**。これは、キューを確認するエージェントのメールアドレスです。複数の受信側を一覧表示するには、メールアドレスをコンマで区切ります。**キューの新しいリクエストのみ。**
   - **Subject**。通知の件名のタイトルを入力します。
   - **コンテンツテンプレートパス。**メッセージコンテンツの作成に使用するテンプレートを含むディレクトリーへのパス (ファイル名を含む) を入力します。

5. **Save** をクリックします。



注記

メールサーバーが正しく設定されていることを確認してください。「証明書システム通知用のメールサーバーの設定」 を参照してください。

6. 通知メッセージのテンプレートをカスタマイズします。詳細は、「通知メッセージのカスタマイズ」 を参照してください。

7. 設定をテストします。「設定のテスト」 を参照してください。

注記

pkiconsole が非推奨になりました。

12.2.2. CS.cfg ファイルを編集して特定の通知を設定

1. CA サブシステムを停止します。

```
pki-server stop instance_name
```

2. そのインスタンスの **CS.cfg** ファイルを開きます。このファイルは、インスタンスの **conf/** ディレクトリー内にあります。

3. 有効にする通知タイプのすべての設定パラメーターを編集します。

   証明書発行の通知には、4 つのパラメーターがあります。

```
ca.notification.certIssued.emailSubject
ca.notification.certIssued.emailTemplate
ca.notification.certIssued.enabled
ca.notification.certIssued.senderEmail
```

   証明書失効リストの通知については、4 つのパラメーターがあります。

```
ca.notification.certRevoked.emailSubject
ca.notification.certRevoked.emailTemplate
ca.notification.certRevoked.enabled
ca.notification.certRevoked.senderEmail
```

   証明書要求通知には、5 つのパラメーターがあります。

```
ca.notification.requestInQ.emailSubject
ca.notification.requestInQ.emailTemplate
ca.notification.requestInQ.enabled
ca.notification.requestInQ.recipientEmail
ca.notification.requestInQ.senderEmail
```

   通知メッセージのパラメーターは、「CA の自動通知の設定」 で説明されています。

4. ファイルを保存します。

5. CA インスタンスを再起動します。

```
pki-server start instance_name
```

6. 自動メッセージを送信するジョブが作成されている場合は、メールサーバーが正しく設定されていることを確認してください。「証明書システム通知用のメールサーバーの設定」 を参照してください。

7. 自動的に送信されたメッセージはカスタマイズ可能です。詳細は 「通知メッセージのカスタマイズ」 を参照してください。

12.2.3. 設定のテスト

サブシステムが設定どおりにメール通知を送信するかどうかをテストするには、以下を行います。

1. キュー通知内の要求の通知設定の電子メールアドレスを、アクセス可能なエージェントまたは管理者のメールアドレスに変更します。

2. エンドエンティティーページを開き、エージェント承認の登録フォームを使用して証明書をリクエストします。

   リクエストがエージェントの承認に対してキューに入れられると、リクエストインキューのメール通知が送信されます。メッセージを確認して、設定された情報が含まれているかどうかを確認します。

3. エージェントインターフェイスにログインし、要求を承認します。

   サーバーが証明書を発行すると、ユーザーは要求にリストされているアドレスに証明書が発行したメール通知を受け取ります。メッセージをチェックして、正しい情報があるかどうかを確認します。

4. エージェントインターフェイスにログインし、証明書を取り消します。

   ユーザーのメールアドレスには、証明書が取り消されたことを示すメッセージが含まれている必要があります。メッセージをチェックして、正しい情報があるかどうかを確認します。

12.3. 通知メッセージのカスタマイズ

メール通知は、各タイプのメッセージに対してテンプレートを使用して構築されます。これにより、メッセージは通知、簡単に再現でき、カスタマイズが簡単に行えます。CA は通知メッセージにテンプレートを使用します。HTML とプレーンテキストメッセージには別々のテンプレートがあります。

12.3.1. CA 通知メッセージのカスタマイズ

それぞれの種類の CA 通知メッセージには、HTML テンプレートとそれに関連するプレーンテキストテンプレートがあります。メッセージは、HTML テンプレート、HTML マークアップ用のテキスト、トークンから構築されます。**Tokens** は、メッセージの作成時に現在の値で置き換えられるメッセージで、ドル記号 (**$**) で識別される変数です。利用可能なトークンの一覧については、表12.3「通知変数」 を参照してください。

メッセージタイプの内容は、メッセージテンプレートのテキストおよびトークンを変更することで変更できます。HTML メッセージの外観は、HTML メッセージテンプレートの HTML コマンドを変更することで変更できます。

certificate-issuance-notification メッセージのデフォルトのテキストバージョンは以下の通りです。

```
Your certificate request has been processed successfully.
SubjectDN= $SubjectDN
IssuerDN= $IssuerDN
notAfter= $NotAfter
notBefore= $NotBefore
Serial Number= 0x$HexSerialNumber
To get your certificate, please follow this URL:
https://$HttpHost:$HttpPort/displayBySerial?op=displayBySerial&
 serialNumber=$SerialNumber
Please contact your admin if there is any problem.
And, of course, this is just a \$SAMPLE\$ email notification form.
```

このテンプレートは、次のように、トークンとテキストを再配置、追加、または削除することにより、必要に応じてカスタマイズできます。

```
THE EXAMPLE COMPANY CERTIFICATE ISSUANCE CENTER
Your certificate has been issued!
You can pick up your new certificate at the following website:
https://$HttpHost:$HttpPort/displayBySerial?op=displayBySerial&
 serialNumber=$SerialNumber
This certificate has been issued with the following information:
Serial Number= 0x$HexSerialNumber
Name of Certificate Holder = $SubjectDN
Name of Issuer = $IssuerDN
Certificate Expiration Date = $NotAfter
Certificate Validity Date = $NotBefore
Contact IT by calling X1234, or going to the IT website http://IT
 if you have any problems.
```

通知メッセージテンプレートは **/var/lib/pki/*instance_name*/ca/emails** ディレクトリーにあります。

これらのメッセージの名前と場所を変更することができます。通知の設定時に適切な変更を加えます。証明書が拒否されたテンプレートを除いて、すべてのテンプレート名を変更できます。これらの名前は同じままにする必要があります。証明書の発行と拒否に関連するテンプレートは、同じディレクトリーに配置し、同じ拡張子を使用する必要があります。

表12.1「通知テンプレート」 には、通知メッセージの作成に提供されたデフォルトのテンプレートファイルを一覧表示します。表12.2「ジョブ通知のメールテンプレート」 は、ジョブ概要メッセージの作成に提供されたデフォルトのテンプレートファイルを一覧表示します。

表12.1 通知テンプレート

ファイル名	説明
certIssued_CA	証明書の発行時のプレーンテキスト通知メールのテンプレート。
certIssued_CA.html	証明書が発行されたときにエンドエンティティーに送信される HTML ベースの通知メールのテンプレート。
certRequestRejected.html	証明書リクエストが拒否される際に、エンドエンティティーに対する HTML ベースの通知メールのテンプレート。
certRequestRevoked_CA	証明書が取り消されたときにエンドエンティティーに送信されるプレーンテキストの通知メールのテンプレート。
certRequestRevoked_CA.html	証明書が取り消されたときにエンドエンティティーに送信される HTML ベースの通知メールのテンプレート。
reqInQueue_CA	リクエストがキューに入ったときのエージェントへのプレーンテキスト通知メールのテンプレート。

ファイル名	説明
reqInQueue_CA.html	リクエストがキューに入ったときのエージェントへの HTML ベースの通知メールのテンプレート。

表12.2 ジョブ通知のメールテンプレート

ファイル名	説明
rnJob1.txt	エンドエンティティーに送信されるメッセージコンテンツを作成して、証明書の有効期限が近づいていること、および証明書の有効期限が切れる前に証明書を更新または置換する必要があることを通知するためのテンプレート。
rnJob1Summary.txt	サマリーレポートをエージェントおよび管理者に送信するためのテンプレート。**rnJob1Item.txt** テンプレートを使用してメッセージ内のアイテムをフォーマットします。
rnJob1Item.txt	サマリーレポートに含まれるアイテムをフォーマットするためのテンプレート。
riq1Item.html	サマリーテーブルに含まれる項目をフォーマットするためのテンプレート。これは、**riq1Summary.html** テンプレートを使用して構築されます。
riq1Summary.html	Certificate Manager のエージェントキューで保留中の要求数を報告するレポートまたはテーブルを計算するテンプレート。
publishCerts	ディレクトリーに公開される証明書を要約したレポートまたはテーブルのテンプレート。**publishCertsItem.html** テンプレートを使用して、テーブルのアイテムをフォーマットします。
publishCertsItem.html	サマリーテーブルに含まれるアイテムをフォーマットするためのテンプレート。
ExpiredUnpublishJob	ディレクトリーに公開される証明書を要約したレポートまたはテーブルのテンプレート。**ExpiredUnpublishJobItem** テンプレートを使用して、テーブルのアイテムをフォーマットします。
ExpiredUnpublishJobItem	サマリーテーブルに含まれるアイテムをフォーマットするためのテンプレート。

表12.3「通知変数」 通知メッセージテンプレートで使用できる変数の一覧表示および定義します。

表12.3 通知変数

トークン	説明
$CertType	証明書のタイプを指定します。次のいずれかになります。 ● TLS クライアント (**client**) ● TLS サーバー (**server**) ● CA 署名証明書 (**ca**) ● その他 (**other**)
$ExecutionTime	ジョブが実行された時間を指定します。
$HexSerialNumber	16 進形式で発行された証明書のシリアル番号を指定します。
$HttpHost	Certificate Manager の完全修飾ホスト名を指定して、証明書を取得するエンドエンティティーが接続します。
$HttpPort	Certificate Manager のエンドエンティティー (TLS 以外の) ポート番号を指定します。
$InstanceID	通知を送信するサブシステムの ID を指定します。
$IssuerDN	証明書を発行した CA の DN を指定します。
$NotAfter	有効期間の終了日を指定します。
$NotBefore	有効期間の開始日を指定します。
$RecipientEmail	受信者のアドレスを指定します。
$RequestId	要求 ID を指定します。
$RequestorEmail	リクエスターのメールアドレスを指定します。
$RequestType	作成されたリクエストのタイプを指定します。
$RevocationDate	証明書が取り消された日付を指定します。
$SenderEmail	送信者のメールアドレスを指定します。これは、通知設定の **Sender の E-mail Address** フィールドで指定されるアドレスと同じです。
$SerialNumber	発行した証明書のシリアル番号を指定します。シリアル番号は、メッセージで 16 進数で表示されます。

トークン	説明
$Status	要求のステータスを指定します。
$SubjectDN	証明書サブジェクトの DN を指定します。
$SummaryItemList	概要通知のアイテムを表示します。各項目は、ジョブがパブリッシュディレクトリーから更新または削除を検出する証明書に対応します。
$SummaryTotalFailure	失敗したサマリーレポートの合計項目数を指定します。
$SummaryTotalNum	キューで保留中の証明書要求の総数、または要約レポートのディレクトリーから更新または削除される証明書の総数を示します。
$SummaryTotalSuccess	サマリーレポートのアイテムの合計数を表示します。

12.4. 証明書システム通知用のメールサーバーの設定

通知およびジョブ機能は、Certificate System CA インスタンスで設定されたメールサーバーを使用して通知メッセージを送信します。

メールサーバーの設定を開始する前に、**CS.cfg** 設定ファイルで以下のパラメーターが指定されていることを確認してください。

```
smtp.host=localhost
smtp.port=25
```

以下の手順を実行してメールサーバーを設定します。

1. CA サブシステム管理コンソールを開きます。以下に例を示します。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **設定** タブで、上部のインスタンス名を強調表示し、**SMTP** タブを選択します。
3. メールサーバーのサーバー名およびポート番号を指定します。

   サーバー名は、メールサーバーがインストールされているマシンの完全修飾 DNS ホスト名です (**mail.example.com**)。デフォルトでは、メールサーバーのホスト名は、実際のホスト名ではなく **localhost** です。

   SMTP メールサーバーがリッスンするデフォルトのポート番号は **25** です。
4. **Save** をクリックします。

注記

pkiconsole が非推奨になりました。

12.5. CA のカスタム通知の作成

Certificate System CA の既存のメール通知プラグインを編集することで、トークン登録などの他の PKI 操作を処理するカスタム通知機能を作成できます。カスタム通知プラグインの作成または使用を試みる前に、Red Hat サポートサービスにお問い合わせください。

第13章 自動ジョブの設定

Certificate System は、**cron** ジョブのスケジュールに対するさまざまなメカニズムに対応するカスタマイズ可能な Job Scheduler を提供します。本章では、ジョブ実行に特定のジョブプラグインモジュールを使用するように Certificate System を設定する方法を説明します。

13.1. 自動ジョブについて

Certificate Manager コンソールには、指定したタイミングで特定のジョブを実行できる *Job Scheduler* オプションが含まれます。Job Scheduler は従来の Unix **cron** デーモンと似ています。登録済みの **cron** ジョブを取得して、事前に設定された日時で実行します。設定されている場合、スケジューラーは指定された間隔で実行を待機しているジョブをチェックします。指定された実行時間に達すると、スケジューラーはジョブを自動的に開始します。

ジョブは Java™ クラスとして実装され、Certificate System にプラグインモジュールとして登録されます。ジョブモジュールの 1 つの実装を使用して、ジョブの複数のインスタンスを設定できます。各インスタンスには一意の名前 (スペースを含まない英数字の文字列) が必要であり、さまざまなジョブに適用するためにさまざまな入力パラメーター値を含めることができます。

13.1.1. 自動ジョブの設定

自動ジョブ機能は、次のようにして設定されます。

- Job Scheduler の有効化および設定。詳細は 「ジョブスケジューラーの設定」 を参照してください。
- ジョブモジュールの有効化および設定と、これらのジョブモジュールの設定を行います。詳細は 「特定のジョブの設定」 を参照してください。
- 通知のタイプに関連付けられたテンプレートを変更して、これらのジョブで送信されるメール通知メッセージをカスタマイズします。メッセージの内容は、プレーンテキストメッセージと HTML メッセージの両方で設定されます。外観は、HTML テンプレートを変更することで変更されます。詳細は、 「CA 通知メッセージのカスタマイズ」 を参照してください。

13.1.2. 自動ジョブの種類

自動ジョブのタイプは、**RenewalNotificationJob**、**RequestInQueueJob**、**PublishCertsJob**、**UnpublishExpiredJob** です。Certificate System のデプロイ時に各ジョブタイプのインスタンスが 1 つ作成されます。

13.1.2.1. certRenewalNotifier (RenewalNotificationJob)

certRenewalNotifier ジョブは、内部データベースで期限切れになる証明書をチェックします。見つかった場合は、証明書の所有者に自動的に電子メールを送信し、設定された期間または証明書が置き換えられるまで、電子メールによるリマインダーを送信し続けます。ジョブはすべての更新通知の概要を収集し、設定されたエージェントまたは管理者に概要を送ります。

ジョブは、メールリゾルバーを使用して通知を送信するメールアドレスを決定します。デフォルトでは、メールアドレスは証明書自体または証明書に関連する登録要求にあります。

13.1.2.2. requestInQueueNotifier (RequestInQueueJob)

requestInQueueNotifier ジョブは、事前に設定された時間間隔で要求キューのステータスを確認します。延期された登録要求がキューで待機している場合、ジョブはその結果を要約した電子メールメッセージを作成し、指定されたエージェントに送信します。

13.1.2.3. publishCerts (PublishCertsJob)

publishCerts ジョブは、まだ公開されていない公開ディレクトリーに追加された新しい証明書をチェックします。これらの新しい証明書が追加されると、その証明書は、**publishCerts** ジョブにより LDAP ディレクトリーまたはファイルに自動的に公開されます。

注記

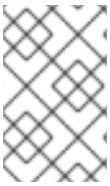
ほとんどの場合、そのルールにマッチした証明書を直ちに適切な公開ディレクトリーに公開します。

証明書が作成時に正常に公開されると、**publishCerts** ジョブは証明書を再公開しません。したがって、サマリーは **publishCerts** ジョブにより公開される証明書のみを一覧表示するため、新しい証明書はジョブサマリーレポートには一覧表示されません。

13.1.2.4. unpublishExpiredCerts (UnpublishExpiredJob)

期限切れの証明書は、公開ディレクトリーから自動的に削除されません。Certificate Manager が LDAP ディレクトリーに証明書を公開するように設定されている場合、ディレクトリーに期限切れの証明書が含まれるようにします。

unpublishExpiredCerts ジョブは、有効期限が切れた証明書をチェックし、設定された間隔で内部データベースで **published** としてマークされます。ジョブはパブリッシュディレクトリーに接続し、これらの証明書を削除します。次に、これらの証明書を内部データベースの **unpublished** としてマークします。ジョブは削除された期限切れの証明書の概要を収集し、設定で指定されたエージェントまたは管理者に概要をメールします。



注記

このジョブは、ディレクトリーから期限切れの証明書の自動削除を自動化します。期限切れの証明書は手動で削除することもできます。詳細は、「ディレクトリーの証明書および CRL の更新」 を参照してください。

13.2. ジョブスケジューラーの設定

Certificate Manager は、Job Scheduler が有効な場合に限りジョブを実行できます。ジョブスケジュールの有効化、周波数の設定、ジョブモジュールの有効化などのジョブ設定は、Certificate System CA Console または **CScfg.** ファイルを編集することで実行できます。

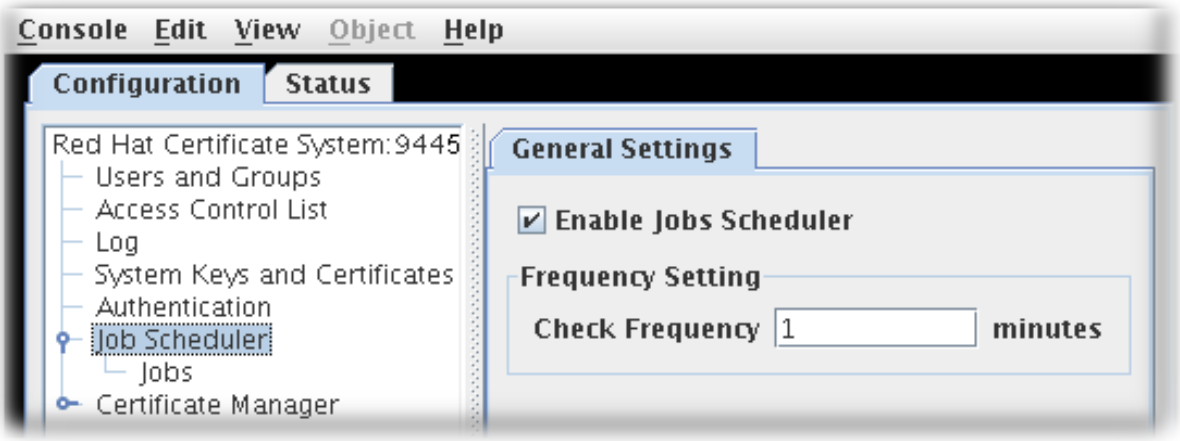
ジョブスケジューラーを有効にするには、次のコマンドを実行します。

1. Certificate Manager Console を開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブナビゲーションツリーで、**Job Scheduler** をクリックします。

   これにより、**General Settings** タブが開き、Job Scheduler が現在有効になっているかどうかを確認します。

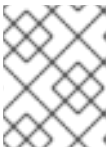


3. **Enable Jobs Schedule** チェックボックスをクリックして、Job Scheduler を有効または無効にします。

   ジョブスケジューラーを無効にすると、すべてのジョブが無効になります。

4. スケジューラーが **Check Frequency** フィールドでジョブをチェックする頻度を設定します。

   頻度は、Job Scheduler デーモンスレッドがウェイクアップし、**cron** 指定に対応する設定済みのジョブを呼び出す頻度です。デフォルトでは、1分に設定されています。



注記

この情報を入力するウィンドウは小さすぎて入力内容を確認できない可能性があります。Certificate Manager Console の隅をウィンドウ全体を拡大します。

5. **Save** をクリックします。

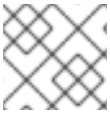
注記

pkiconsole が非推奨になりました。

13.3. 特定のジョブの設定

自動ジョブは、Certificate Manager Console を使用するか、設定ファイルディレクトリーを編集して設定できます。これらの変更は、Certificate Manager Console から行うことが推奨されます。

13.3.1. 証明書マネージャーコンソールを使用した特定のジョブの設定



注記

pkiconsole が非推奨になりました。

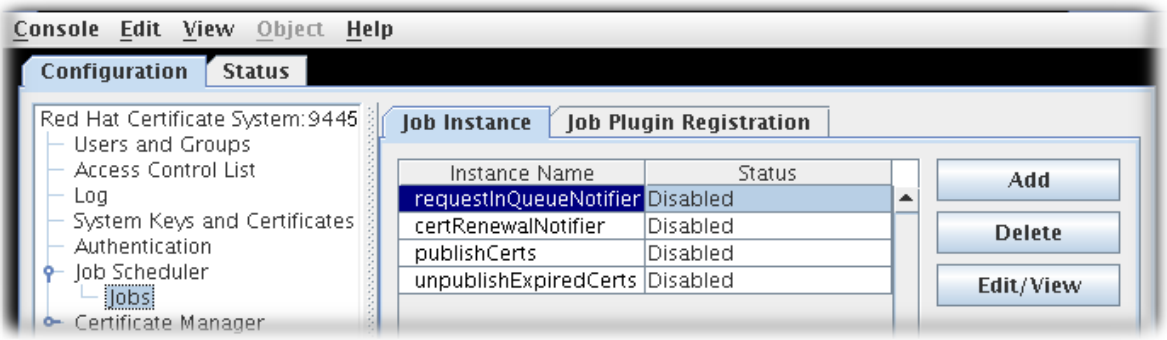
Certificate Manager コンソールを使用して自動ジョブを有効化して設定するには、以下を実行します。

1. Certificate Manager Console を開きます。

```
pkiconsole https://server.example.com:8443/ca
```

2. ジョブスケジューラーが有効になっていることを確認します。詳細は、「ジョブスケジューラーの設定」 を参照してください。

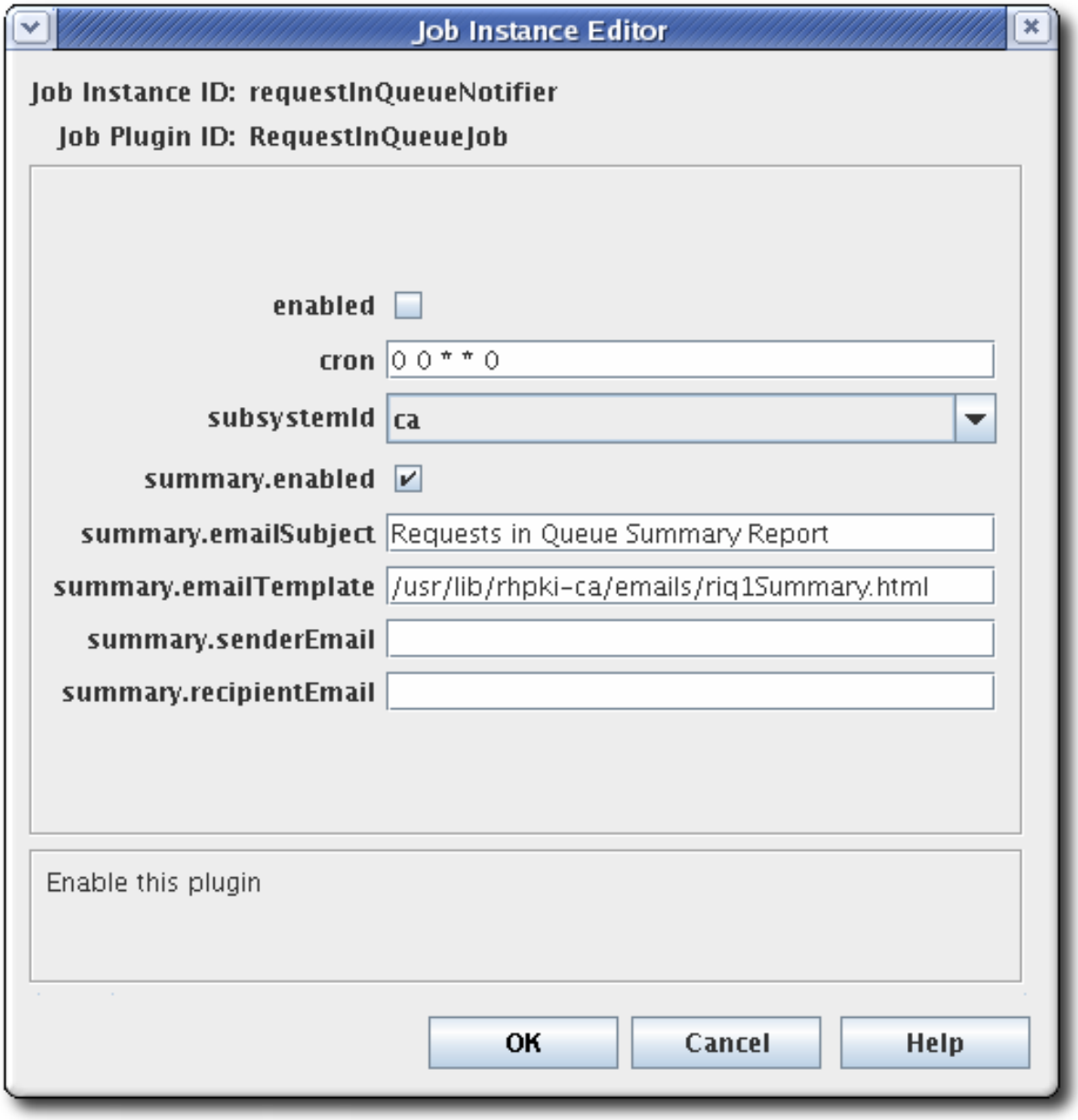
3. **Configuration** タブで、ナビゲーションツリーから **Job Scheduler** を選択します。次に **Jobs** を選択して、**Job Instance** タブを開きます。



一覧からジョブインスタンスを選択し、**Edit/View** をクリックします。

ジ**Job Instance Editor** が開き、現在のジョブ設定が表示されます。

図13.1 ジョブ設定



4. ジョブを有効にするには **enabled** を選択します。
5. このダイアログのフィールドで設定設定を指定して設定します。
   - **certRenewalNotifier** は、「certRenewalNotifier の設定パラメーター」 を参照してください。
   - **requestInQueueNotifier** は、「requestInQueueNotifier の設定パラメーター」 を参照してください。
   - **publishCerts** は、「publishCerts の設定パラメーター」 を参照してください。
   - **unpublishExpiredCerts** は、「unpublishExpiredCerts の設定パラメーター」 を参照してください。
   - **cron** 時間頻度の設定に関する詳細は、「自動ジョブの頻度設定」 を参照してください。
6. **OK** をクリックします。

7. **Refresh** をクリックしてメインのウィンドウで変更を表示します。

8. ジョブが自動メッセージを送信するように設定されている場合は、メールサーバーが正しく設定されていることを確認してください。「証明書システム通知用のメールサーバーの設定」 を参照してください。

9. 電子メールメッセージテキストと外観をカスタマイズします。

13.3.2. 設定ファイルを編集してジョブを設定

1. ジョブスケジューラーが有効で設定されていることを確認します。「ジョブスケジューラーの設定」 を参照してください。

2. CA サブシステムインスタンスを停止します。

```
pki-server stop instance_name
```

3. テキストエディターで、そのサーバーインスタンスの **CS.cfg** ファイルを開きます。

4. 設定されているジョブモジュールのすべての設定パラメーターを編集します。

   - **certRenewalNotifier** ジョブを設定するには、**jobsScheduler.job.certRenewalNotifier** で始まるすべてのパラメーターを編集します。「certRenewalNotifier の設定パラメーター」 を参照してください。

   - **requestInQueueNotifier** ジョブを設定するには、**jobsScheduler.job.requestInQueueNotifier** で始まるすべてのパラメーターを編集します。「requestInQueueNotifier の設定パラメーター」 を参照してください。

   - **publishCerts** ジョブを設定するには、**jobsScheduler.job.publishCerts** で始まるすべてのパラメーターを編集します。「publishCerts の設定パラメーター」 を参照してください。

   - **unpublishExpiredCerts** ジョブを設定するには、**jobsScheduler.job.unpublishExpiredCerts** で始まるすべてのパラメーターを編集します。「unpublishExpiredCerts の設定パラメーター」 を参照してください。

5. ファイルを保存します。

6. サーバーインスタンスを再起動します。

```
pki-server start instance_name
```

7. ジョブが自動的にメッセージを送信する場合は、メールサーバーが正しく設定されていることを確認してください。「証明書システム通知用のメールサーバーの設定」 を参照してください。

8. 自動ジョブメッセージをカスタマイズします。

13.3.3. certRenewalNotifier の設定パラメーター

表13.1「certRenewalNotifier パラメーター」 **CS.cfg** ファイルまたは Certificate Manager コンソール のいずれかで、**certRenewalNotifier** ジョブに設定できるこれらのパラメーターの詳細を提供します。

表13.1 certRenewalNotifier パラメーター

パラメーター	説明
enabled	ジョブを有効または無効にするかどうかを指定します。**true** の場合はジョブを有効にします。**false** に設定すると 無効にします。
cron	このジョブの実行スケジュールを設定します。これにより、Job Scheduler デーモンスレッドが、更新通知を送信するために証明書をチェックする時間を設定します。これらの設定は、「自動ジョブの頻度設定」 の規則に従う必要があります。以下に例を示します。 `0 3 * * 1-5` この例のジョブは、月曜日から金曜日の午後 3 時まで実行されます。
notifyTriggerOffset	証明書の有効期限の前に最初の通知が送信される期間 (日数) を設定します。
notifyEndOffset	証明書の有効期限が切れてから、証明書が置き換えられない場合に通知が送信され続ける期間 (日数) を設定します。
senderEmail	配信問題について通知する通知メッセージの送信者を設定します。
emailSubject	通知メッセージの Subject 行のテキストを設定します。
emailTemplate	メッセージコンテンツの作成に使用するテンプレートが含まれるディレクトリーに、ファイル名を含むパスを設定します。
summary.enabled	更新通知の概要レポートをコンパイルして送信すべきかどうかを設定します。**true** の値はサマリーを送信できるようにします。**false** はこれを無効にします。有効にする場合は、残りのサマリーパラメーターを設定します。これは、サーバーでサマリーレポートを送信するために必要です。
summary.recipientEmail	サマリーメッセージの受信者を指定します。これらは、ユーザー証明書または他のユーザーのステータスを知る必要があるエージェントである可能性があります。各メールアドレスをコンマで区切ることで、複数の受信者を設定できます。
summary.senderEmail	サマリーメッセージの送信者のメールアドレスを指定します。
summary.emailSubject	要約メッセージの件名を指定します。

パラメーター	説明
summary.itemTemplate	サマリーレポート用に収集される各アイテムのコンテンツおよび形式を作成するために使用するテンプレートが含まれるディレクトリーに、ファイル名を含むパスを指定します。
summary.emailTemplate	サマリーレポートのメール通知を作成するために使用するテンプレートが含まれるディレクトリーに、ファイル名を含むパスを指定します。

13.3.4. requestInQueueNotifier の設定パラメーター

表13.2「requestInQueueNotifier パラメーター」 **CS.cfg** ファイルまたは Certificate Manager コンソール のいずれかで、**requestInQueueNotifier** ジョブに設定できるこれらのパラメーターの詳細を提供します。

表13.2 requestInQueueNotifier パラメーター

パラメーター	説明
enabled	ジョブを有効 (**true**) または無効 (**false**) にするかを設定します。
cron	ジョブを実行する時刻を設定します。これは、Job Scheduler デーモンスレッドが保留中のリクエストのキューをチェックする時間です。この設定は、「自動ジョブの頻度設定」 の規則に従う必要があります。以下に例を示します。 `0 0 * * 0`
subsystemid	ジョブを実行しているサブシステムを指定します。Certificate Manager で利用できる値は **ca** です。
summary.enabled	達成されたジョブの要約をコンパイルして送信するかどうかを指定します。**true** 値によりサマリーレポートが有効になり、**false** により無効になります。有効にする場合は、残りのサマリーパラメーターを設定します。これは、サーバーでサマリーレポートを送信するために必要です。
summary.emailSubject	要約メッセージの件名を指定します。
summary.emailTemplate	要約レポートの作成に使用するテンプレートを含むディレクトリーへのパス (ファイル名を含む) を指定します。
summary.senderEmail	配信問題について通知する通知メッセージの送信者を指定します。

パラメーター	説明
summary.recipientEmail	サマリーメッセージの受信者を指定します。これらは、保留中の要求または他のユーザーを処理する必要があるエージェントである可能性があります。各メールアドレスをコンマで区切ることで、複数の受信者を一覧に追加することができます。

13.3.5. publishCerts の設定パラメーター

表13.3「publishCerts パラメーター」 **CS.cfg** ファイルまたは Certificate Manager コンソール のいずれかで、**publishCerts** ジョブに設定できるこれらのパラメーターの詳細を提供します。

表13.3 publishCerts パラメーター

パラメーター	説明
enabled	ジョブが有効かどうかを指定します。**true** の値は有効で、**false** 無効になります。
cron	ジョブの実行時には、時間スケジュールを設定します。これは、Job Scheduler デーモンスレッドが証明書をチェックして、公開ディレクトリーから期限切れの証明書を削除する時間です。この設定は、「自動ジョブの頻度設定」 の規則に従う必要があります。以下に例を示します。 `0 0 * * 6`
summary.enabled	ジョブによって公開される証明書の概要をコンパイルおよび送信するかどうかを指定します。**true** 値によりサマリーレポートが有効になり、**false** により無効になります。有効にする場合は、残りのサマリーパラメーターを設定します。これは、サーバーでサマリーレポートを送信するために必要です。
summary.emailSubject	要約メッセージの件名を指定します。
summary.emailTemplate	要約レポートの作成に使用するテンプレートを含むディレクトリーへのパス (ファイル名を含む) を指定します。
summary.itemTemplate	ファイル名を含むパスを指定し、サマリーレポート用に収集された各アイテムのコンテンツおよび形式を作成するのに使用するテンプレートが含まれるディレクトリーへのパスを指定します。
summary.senderEmail	配信の問題について通知するサマリーメッセージの送信者を指定します。

パラメーター	説明
summary.recipientEmail	サマリーメッセージの受信者を指定します。これらは、ユーザー証明書または他のユーザーのステータスを知る必要があるエージェントである可能性があります。各メールアドレスをコンマで区切ることで、複数の受信者を設定できます。

13.3.6. unpublishExpiredCerts の設定パラメーター

表13.4「unpublishExpiredCerts パラメーター」 **CS.cfg** ファイルまたは Certificate Manager コンソールのいずれかで、**unpublishedExpiresCerts** ジョブに設定できるこれらのパラメーターの詳細を提供します。

表13.4 unpublishExpiredCerts パラメーター

パラメーター	説明
enabled	ジョブが有効かどうかを指定します。**true** の値は有効で、**false** 無効になります。
cron	ジョブの実行時には、時間スケジュールを設定します。これは、Job Scheduler デーモンスレッドが証明書をチェックして、公開ディレクトリーから期限切れの証明書を削除する時間です。この設定は、「自動ジョブの頻度設定」 の規則に従う必要があります。以下に例を示します。 `0 0 * * 6`
summary.enabled	ジョブによって公開される証明書の概要をコンパイルおよび送信するかどうかを指定します。**true** 値によりサマリーレポートが有効になり、**false** により無効になります。有効にする場合は、残りのサマリーパラメーターを設定します。これは、サーバーでサマリーレポートを送信するために必要です。
summary.emailSubject	要約メッセージの件名を指定します。
summary.emailTemplate	要約レポートの作成に使用するテンプレートを含むディレクトリーへのパス (ファイル名を含む) を指定します。
summary.itemTemplate	ファイル名を含むパスを指定し、サマリーレポート用に収集された各アイテムのコンテンツおよび形式を作成するのに使用するテンプレートが含まれるディレクトリーへのパスを指定します。
summary.senderEmail	配信の問題について通知するサマリーメッセージの送信者を指定します。

パラメーター	説明
summary.recipientEmail	サマリーメッセージの受信者を指定します。これらは、ユーザー証明書または他のユーザーのステータスを知る必要があるエージェントである可能性があります。各メールアドレスをコンマで区切ることで、複数の受信者を設定できます。

13.3.7. 自動ジョブの頻度設定

Job Scheduler は Unix **crontab** エントリー形式のバリエーションを使用して、ジョブキューをチェックしてジョブを実行する日時を指定します。表13.5「ジョブのスケジュール設定の時間値」 および 図13.1「ジョブ設定」 にあるように、時間エントリーの形式は 5 つのフィールドで設定されます。(Unix **contab** が指定された 6 番目のフィールドは Job Scheduler で使用されません。) 値はスペースまたはタブで区切られます。

単一の整数またはハイフン (**-**) で区切られた整数のペアのいずれかを含めて、包括的範囲を示すことができます。すべての有効な値を指定するには、フィールドに整数ではなくアスタリスクを含めることができます。日フィールドには、値のコンマ区切りリストを含めることができます。この式の構文は、以下のようになります。

> *Minute Hour Day_of_month Month_of_year Day_of_week*

表13.5 ジョブのスケジュール設定の時間値

フィールド	値
Minute	0-59
Hour	0-23
Day of month	1-31
Month of year	1-12
Day of week	0-6 (0= 日曜日)

たとえば、以下の時間エントリーは毎時 15 分 (1:15、2:15、3:15 など) を指定します。

```
15 * * * *
```

次の例では、4 月 12 日の正午に実行するジョブを設定します。

```
0 12 12 4 *
```

day-of-month および day-of-week オプションには、複数の日を指定するための値のコンマ区切りリストを含めることができます。両日フィールドを指定すると、仕様が含められます。その日は、有効な曜日に追記する必要がありません。たとえば、次のエントリーは、毎月 1 日 と 15 日、**および** 毎週月曜日の深夜にジョブ実行時間を指定します。

```
0 0 1,15 * 1
```

ある日のタイプを他の日付を使用せずに指定するには、その他の日付フィールドにアスタリスクを使用します。たとえば、以下のエントリーは、平日の午前 3 時 15 分にジョブを実行します。

```
15 3 * * 1-5
```

13.4. ジョブモジュールの登録

カスタムジョブプラグインは、Certificate Manager コンソールから登録できます。新しいモジュールを登録するには、モジュール名と、モジュールを実装する Java™ クラスのフルネームを指定する必要があります。

新規ジョブモジュールを登録するには、次のコマンドを実行します。

1. カスタムジョブクラスを作成します。この例では、カスタムジョブプラグインは **MyJob.java** と呼ばれます。
2. 新しいクラスをコンパイルします。

   ```
   javac -d . -classpath $CLASSPATH MyJob.java
   ```

3. CA がカスタムクラスにアクセスできるように、CA の **WEB-INF** Web ディレクトリーにディレクトリーを作成します。

   ```
   mkdir /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
   ```

4. 新しいプラグインファイルを新しい **class** ディレクトリーにコピーし、所有者を Certificate System system user (**pkiuser**) に設定します。

   ```
   cp -pr com /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes

   chown -R pkiuser:pkiuser /var/lib/pki/instance_name/ca/webapps/ca/WEB-INF/classes
   ```

5. プラグインを登録します。
   a. Certificate Manager コンソールにログインします。

      ```
      pkiconsole https://server.example.com:8443/ca
      ```

   b. **Configuration** タブで、左側のナビゲーションツリーで **Job Scheduler** を選択します。**Jobs** を選択します。

      ジョブインスタンスタブが開き、現在設定されているジョブが一覧表示されます。**Job Plugin Registration** タブを選択します。

   c. **Register** をクリックして、新しいモジュールを追加します。

   d. **Register Job Scheduler Plugin Implementation** ウィンドウで、以下の情報を入力します。

      - **プラグイン名。** プラグインモジュールの名前を入力します。

- **クラス名。**このモジュールのクラスのフルネームを入力します。これは実装する Java™ クラスへのパスです。このクラスがパッケージに含まれる場合は、パッケージ名を含めます。たとえば、**com.customplugins** という名前のパッケージに含まれる **customJob** という名前のクラスを登録するには、**com.customplugins.customJob** と入力します。

e. **OK** をクリックします。

注記

ジョブモジュールを削除することもできますが、これは推奨されません。

モジュールを削除する必要がある場合は、新規モジュールの登録時に **Job Plugin Registration** タブを開き、削除するモジュールを選択し、**Delete** をクリックします。プロンプトが表示されたら、削除を確定します。

注記

pkiconsole が非推奨になりました。

パート IV. サブシステムインスタンスの管理

第14章 基本的なサブシステム管理

本章では、Certificate System の管理コンソール、設定ファイル、およびサーバーの起動と停止、ログの管理、ポート割り当ての変更、内部データベースの変更など、その他の基本的な管理タスクについて説明します。

14.1. PKI インスタンス

このバージョンの Certificate System は、すべてのサブシステムで *個別の PKI インスタンス* を引き続きサポートします。

個別の PKI インスタンス

- 単一の Java ベースの Apache Tomcat インスタンスとして実行します。
- 単一の PKI サブシステム (CA、KRA、OCSP、TKS、または TPS) が含まれます。
- 同じ物理マシンまたは仮想マシン (VM) に共存する場合は、一意のポートを使用する必要があります。

さらに、このバージョンの Certificate System で、*共有 PKI インスタンス* の概念が導入されました。

共有 PKI インスタンス

- 単一の Java ベースの Apache Tomcat インスタンスとして実行します。
- 個別の PKI インスタンスと同一の単一の PKI サブシステムを含めることができます。
- PKI サブシステムの各タイプの 1 つまで、任意の組み合わせを含めることができます。
  - CA
  - TKS
  - CA、KRA
  - CA、OCSP
  - TKS、TPS
  - CA、KRA、TKS、TPS
  - CA、KRA、OCSP、TKS、TPS
  - などになります。
- そのインスタンスに含まれるすべてのサブシステムが同じポートを共有できるようにし、
- 複数の同一物理マシンまたは仮想マシンに共存する場合、一意のポートを使用する必要があります。

14.2. PKI インスタンス実行管理

PKI インスタンスの起動、停止、再起動、または取得の挙動は、実行管理と呼ばれます。各 PKI インスタンス (個別または共有) は、起動、停止、再起動、およびステータスは別々に取得されています。本セクションでは、PKI インスタンスの実行管理について説明します。

14.2.1. PKI インスタンスの起動、停止、および再起動

PKI インスタンスは、**systemd** を使用して、他のシステムプログラムと同様に起動、停止、および再起動します。

1. **root** としてサーバーマシンにログインします。
2. アクションとインスタンス名を指定して、**systemctl** コマンドを実行します。

```
systemctl start|stop|restart pki-tomcatd@instance_name.service
```

以下に例を示します。

```
systemctl restart pki-tomcatd@pki-tomcat.service
```

3. または **pki-server** エイリアスを使用できます。

```
pki-server start|stop|restart instance_name
```

以下に例を示します。

```
pki-server restart pki-tomcat
```

14.2.2. マシン再起動後の PKI インスタンスの再起動

1 つ以上の PKI インスタンスを実行しているコンピューターが予期せずシャットダウンした場合、HTML サービスページと管理コンソールの両方からサブシステムを使用できるようにするには、PKI インスタンスだけでなく多くのサービスを適切な順序で再起動する必要があります。

1. サブシステムで使用される Directory Server インスタンスがローカルマシンにインストールされている場合は、管理サーバーと Directory Server プロセスを再起動します。

```
systemctl start dirsrv-admin.service
systemctl start dirsrv@instance_name.service
```

2. Certificate System サブシステムインスタンスを起動します。

```
pki-server start instance_name
```

14.2.3. PKI インスタンスステータスの確認

この **systemctl** コマンドは、プロセスのステータスを確認し、実行中か、または停止しているかどうかを確認できます。以下に例を示します。

```
systemctl -l status pki-tomcatd@pki-tomcat.service
pki-tomcatd@pki-tomcat.service - PKI Tomcat Server pki-tomcat
   Loaded: loaded (/lib/systemd/system/pki-tomcatd@.service; enabled)
   Active: inactive (dead) since Fri 2015-11-20 19:04:11 MST; 12s ago
```

```
  Process: 8728 ExecStop=/usr/libexec/tomcat/server stop (code=exited, status=0/SUCCESS)
  Process: 8465 ExecStart=/usr/libexec/tomcat/server start (code=exited, status=143)
  Process: 8316 ExecStartPre=/usr/bin/pkidaemon start tomcat %i (code=exited, status=0/SUCCESS)
 Main PID: 8465 (code=exited, status=143)

Nov 20 19:04:10 pki.example.com server[8728]: options used: -Dcatalina.base=/var/lib/pki/pki-tomcat
-Dcatalina.home=/usr/share/tomcat -Djava.endorsed.dirs= -Djava.io.tmpdir=/var/lib/pki/pki-
tomcat/temp -Djava.util.logging.config.file=/var/lib/pki/pki-tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
Nov 20 19:04:10 pki.example.com server[8728]: arguments used: stop
Nov 20 19:04:11 pki.example.com server[8465]: Nov 20, 2015 7:04:11 PM
org.apache.catalina.core.StandardServer await
Nov 20 19:04:11 pki.example.com server[8465]: INFO: A valid shutdown command was received via
the shutdown port. Stopping the Server instance.
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[before_stop]
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[stop]
Nov 20 19:04:11 pki.example.com server[8465]: PKIListener:
org.apache.catalina.core.StandardServer[configure_stop]
Nov 20 19:04:11 pki.example.com server[8465]: Nov 20, 2015 7:04:11 PM
org.apache.coyote.AbstractProtocol pause
Nov 20 19:04:11 pki.example.com server[8465]: INFO: Pausing ProtocolHandler ["http-bio-8080"]
Nov 20 19:04:11 pki.example.com systemd[1]: Stopped PKI Tomcat Server pki-tomcat.
```

インスタンスが実行中の場合、ステータスチェックは次の例のような情報を返します。

```
systemctl -l status pki-tomcatd@pki-tomcat.service
pki-tomcatd@pki-tomcat.service - PKI Tomcat Server pki-tomcat
   Loaded: loaded (/lib/systemd/system/pki-tomcatd@.service; enabled)
   Active: active (running) since Fri 2015-11-20 19:09:09 MST; 3s ago
  Process: 8728 ExecStop=/usr/libexec/tomcat/server stop (code=exited, status=0/SUCCESS)
  Process: 9154 ExecStartPre=/usr/bin/pkidaemon start tomcat %i (code=exited, status=0/SUCCESS)
 Main PID: 9293 (java)
   CGroup: /system.slice/system-pki\x2dtomcatd.slice/pki-tomcatd@pki-tomcat.service
           ├─9293 java -DRESTEASY_LIB=/usr/share/java/resteasy-base -
Djava.library.path=/usr/lib64/nuxwdog-jni -classpath
/usr/share/tomcat/bin/bootstrap.jar:/usr/share/tomcat/bin/tomcat-juli.jar:/usr/share/java/commons-
daemon.jar -Dcatalina.base=/var/lib/pki/pki-tomcat -Dcatalina.home=/usr/share/tomcat -
Djava.endorsed.dirs= -Djava.io.tmpdir=/var/lib/pki/pki-tomcat/temp -
Djava.util.logging.config.file=/var/lib/pki/pki-tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Djava.security.manager -
Djava.security.policy==/var/lib/pki/pki-tomcat/conf/catalina.policy
org.apache.catalina.startup.Bootstrap start

Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.core.StandardService startInternal
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Starting service Catalina
Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.core.StandardEngine startInternal
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Starting Servlet Engine: Apache
Tomcat/7.0.54
Nov 20 19:09:10 pki.example.com server[9293]: Nov 20, 2015 7:09:10 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:10 pki.example.com server[9293]: INFO: Deploying configuration descriptor /etc/pki/pki-
tomcat/Catalina/localhost/ROOT.xml
```

```
Nov 20 19:09:12 pki.example.com server[9293]: Nov 20, 2015 7:09:12 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:12 pki.example.com server[9293]: INFO: Deployment of configuration descriptor
/etc/pki/pki-tomcat/Catalina/localhost/ROOT.xml has finished in 2,071 ms
Nov 20 19:09:12 pki.example.com server[9293]: Nov 20, 2015 7:09:12 PM
org.apache.catalina.startup.HostConfig deployDescriptor
Nov 20 19:09:12 pki.example.com server[9293]: INFO: Deploying configuration descriptor /etc/pki/pki-
tomcat/Catalina/localhost/pki#admin.xml
```

14.2.4. 再起動時に自動的に起動するための PKI インスタンスの設定

systemctl コマンドを使用すると、再起動時にインスタンスを自動的に起動できます。たとえば、以下のコマンドは、再起動後に Red Hat 管理サーバー、Directory Server、および CA を自動的に起動します。

```
# systemctl enable dirsrv-admin.service
# systemctl enable dirsrv.target
# systemctl enable pki-tomcatd@pki-tomcat.service
```

注記

この **pkispawn** コマンドを使用したデフォルトの PKI インスタンスのインストールと設定により、リブート時にインスタンスが自動的に起動できるようになります。

この動作を無効にするには (つまり、再起動後に PKI インスタンスが自動的に起動しないようにするには)、次のコマンドを実行します。

```
# systemctl disable pki-tomcatd@pki-tomcat.service
# systemctl disable dirsrv.target
# systemctl disable dirsrv-admin.service
```

14.2.5. Certificate System Service の sudo パーミッションの設定

管理とセキュリティーの両方を簡素化するために、Certificate System と Directory Server のプロセスを設定して、(root だけでなく) PKI 管理者がサービスを開始および停止できるようにすることができます。

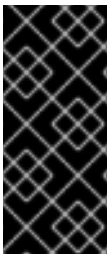
サブシステムを設定する際に **pkiadmin** システムグループの使用が推奨されるオプションです。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』を参照してください。 Certificate System 管理者であるすべてのオペレーティングシステムユーザーがこのグループに追加されます。**pkiadmin** のシステムグループが存在する場合は、特定のタスクを実行するための sudo アクセスを付与することができます。

1. **/etc/sudoers** ファイルを編集します。Red Hat Enterprise Linux 8 では、 **visudo** コマンドを使用して実行できます。

   ```
   # visudo
   ```

2. マシンにインストールされているものに応じて、Directory Server、Administration Server、PKI 管理ツール、および各 PKI サブシステムインスタンスの行を追加して、**pkiadmin** グループに **sudo** 権限を付与します。

```
# For Directory Server services
%pkiadmin ALL = PASSWD: /usr/bin/systemctl * dirsrv.target
%pkiadmin ALL = PASSWD: /usr/bin/systemctl * dirsrv-admin.service

# For PKI instance management
%pkiadmin ALL = PASSWD: /usr/sbin/pkispawn *
%pkiadmin ALL = PASSWD: /usr/sbin/pkidestroy *

# For PKI instance services
%pkiadmin ALL = PASSWD: /usr/bin/systemctl * pki-tomcatd@instance_name.service
```



重要

マシン上のすべての Certificate System、Directory Server、および Administration Server に対して、またマシン上のそれらのインスタンスに対して **のみ**、sudo パーミッションを設定してください。マシンに同じサブシステムタイプのインスタンスが複数存在する場合と、サブシステムタイプのインスタンスが存在しない場合があります。これはデプロイメントによって異なります。

14.3. サブシステムのコンソールおよびサービスを開く

各サブシステムには、ユーザータイプごとに異なるインターフェイスがあります。すべてのサブシステムには、TKS を除くエージェント、管理者、またはエンドユーザー (すべて 3 つ) の Web サービスページがあります。さらに、CA、KRA、OCSP、および TKS はすべて、サーバーにインストールされ、サブシステム自体を管理する管理タスクを実行する Java ベースのコンソールがあります。

外観と、サブシステムの Web ベースのサービスページの機能をカスタマイズして、組織の既存の Web サイトとの統合を改善できます。『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』を参照してください。

14.3.1. サブシステムの Web サービスページの検索

CA、KRA、OCSP、TKS、および TPS サブシステムには、エージェント、通常のユーザー、および管理者向けの Web サービスページがあります。これらの Web サービスのメニューには、サブシステムのセキュアなユーザーのポート上でサブシステムホストに URL を開くとアクセスできます。CA の場合の例を以下に示します。

```
https://server.example.com:8443/ca/services
```

各サブシステムの主な Web サービスページには、利用可能なサービスページの一覧があります。これらは 表14.1「デフォルトの Web サービスページ」 で要約されています。特定のサービスにアクセスするには、適切なポートにアクセスし、適切なディレクトリーを URL に追加します。たとえば、CA のエンドエンティティー (通常のユーザー) の Web サービスにアクセスするには、以下を実行します。

```
https://server.example.com:8443/ca/ee/ca
```

DNS が適切に設定されていると、IPv4 アドレスまたは IPv6 アドレスを使用して、サービスページに接続できます。以下に例を示します。

```
https://1.2.3.4:8443/ca/services
https://[00:00:00:00:123:456:789:00:]:8443/ca/services
```

一部のサブシステムインターフェイスは、それらにアクセスするためにクライアント認証を必要としま

す。通常、インターフェイスはエージェントまたは管理者のロールに関連付けられています。他のインターフェイスは、セキュア (SSL 接続) で実行されるインターフェイスであっても、クライアント認証を必要としません。これらのインターフェイス (エンドエンティティーサービスなど) の一部は、クライアント認証を必要とするように設定できますが、クライアント認証をサポートするように設定することはできません。これらの相違点は、表14.1「デフォルトの Web サービスページ」 に記載されています。

注記

サブシステムのエンドユーザーページには誰でもアクセスできますが、エージェントまたは管理者の Web サービスページにアクセスするには、エージェントまたは管理者の証明書を発行して Web ブラウザーにインストールする必要があります。そうしないと、Web サービスへの認証が失敗します。

表14.1 デフォルトの Web サービスページ

SSL に使用	クライアント認証に使用[a]	Web サービス	Web サービスの場所
Certificate Manager			
No		エンドエンティティー	ca/ee/ca/
はい	いいえ	エンドエンティティー	ca/ee/ca
はい	はい	エージェント	ca/agent/ca
はい	いいえ	サービス	ca/services
はい	いいえ	コンソール	pkiconsole https://**host:port**/ca
キーリカバリー認証局			
はい	はい	エージェント	kra/agent/kra
はい	いいえ	サービス	kra/services
はい	いいえ	コンソール	pkiconsole https://**host:port**/kra
オンライン証明書ステータスマネージャー			
はい	はい	エージェント	ocsp/agent/ocsp
はい	いいえ	サービス	ocsp/services
はい	いいえ	コンソール	pkiconsole https://**host:port**/ocsp

SSL に使用	クライアント認証に使用[a]	Web サービス	Web サービスの場所
トークンキーサービス			
はい	いいえ	サービス	tks/services
はい	いいえ	コンソール	pkiconsole https://**host:port**/tks
トークン処理システム			
はい		サービス	index.cgi

[a] クライアント認証値が **No** のサービスは、クライアント認証を要求するように再設定できます。**Yes** または **No** のいずれかの値を持たないサービスは、クライアント認証を使用するように設定することはできません。

14.3.2. 証明書システム管理コンソールの起動

重要

pkiconsole が非推奨になりました。

pkiconsole コマンドを使用して SSL ポート経由でサブシステムインスタンスに接続すると、コンソールが開きます。このコマンドの形式は以下のとおりです。

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

subsystem_type は **ca**、**kra**、**ocsp**、または **tks** にすることができます。たとえば、これにより KRA コンソールが開きます。

```
pkiconsole https://server.example.com:8443/kra
```

DNS が設定されていない場合は、IPv4 アドレスまたは IPv6 アドレスを使用してコンソールに接続できます。以下に例を示します。

```
pkiconsole https://1.2.3.4:8443/ca
pkiconsole https://[00:00:00:00:123:456:789:00:]:8443/ca
```

14.3.3. Java 管理コンソールの SSL の有効化

Certificate System Console への証明書ベースの認証を有効にして、管理者が Certificate System Console にログインする前にクライアント証明書を使用して認証する必要があります。証明書ベースの認証を有効にする前に、管理者の証明書を保存します。

コンソールで SSL を有効にするには、クライアントとサーバーの両方を設定します。

重要

CA が管理ポートを介したクライアント認証用に設定されていて、その CA がセキュリティードメインマネージャーである場合、**その CA をセキュリティードメインに使用する新しい PKI サブシステムを設定することはできません。**新しい PKI インスタンスは、クライアント認証を使用せずに、管理ポートを介してセキュリティードメイン CA に登録します。CA でクライアント認証が必要な場合、登録は失敗します。

まず、SSL クライアント認証を使用するように Certificate System サーバーを設定します。

1. このシステムを使用して管理者の証明書を保存します。証明書は、CA 自体からのものか、サブシステムの証明書に署名した CA からのものである必要があります。

   a. サブシステムコンソールを開きます。

   b. 左側の **ユーザーとグループ** オプションを選択します。

   c. **ユーザー** タブで管理ユーザーを選択し、**Manage Certificates** をクリックします。

   d. **Import** をクリックします。

   e. Web ブラウザーに保存されている管理者証明書などの base-64 でエンコードされた SSL クライアント証明書を貼り付けます。

   クライアント証明書が SSL クライアント認証に適していることを確認してください。それ以外の場合、サーバーはクライアント証明書を受け入れず、**/var/log/instanceID/system** のエラーログにエラーメッセージを投稿します。

   ```
   failure (14290): Error receiving connection
   SEC_ERROR_INADEQUATE_CERT_TYPE - Certificate type not approved for application.)
   ```

2. サブシステムを停止します。

   ```
   pki-server stop instance_name
   ```

3. インスタンス設定ディレクトリー **/var/lib/pki/*instance_name*/*subsystem_type*/conf** を開きます。

4. **CS.cfg** ファイルを開きます。

5. **authType** パラメーターの値を **pwd** から **sslclientauth** に変更します。

   ```
   authType=sslclientauth
   ```

6. ファイルを保存します。

7. **server.xml** ファイルを開きます。

8. admin インターフェイスコネクターセクションで **clientAuth="false"** 属性を **clientAuth="want"** に変更します。

   ```
   <Connector port="8443" maxHttpHeaderSize="8192"
       maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
       enableLookups="false" disableUploadTimeout="true"
       acceptCount="100" scheme="https" secure="true"
   ```

```
        clientAuth="want" sslProtocol="SSL"
.....
        serverCertFile="/var/lib/pki/pki-tomcat/conf/serverCertNick.conf"
        passwordFile="/var/lib/pki/pki-tomcat/conf/password.conf"
        passwordClass="org.apache.tomcat.util.net.jss.PlainPasswordFile"
        certdbDir="/var/lib/pki/pki-tomcat/alias"/>
```

この **want** 値は、クライアント認証が優先されますが、必須ではありません。これにより、(コンソールなど) 簡単に使用できるインターフェイスを介したクライアント認証が可能になりますが、クライアント認証を簡単にサポートしないクライアント (セキュリティードメイン内の他のサブシステム) は通常の接続を使用して接続できます。

9. サブシステムを起動します。

```
pki-server start instance_name
```

サーバーの設定後、クライアントが SSL クライアント認証を使用するように設定します。

コンソールは、サーバーへの SSL クライアント認証に使用される管理者証明書およびキーにアクセスできる必要があります。コンソールのデフォルト証明書およびキーデータベースは、**.redhat-idm-console** ディレクトリーに保存されます。

管理者証明書およびキーへのアクセスを提供するには、管理者のブラウザーから **.p12** ファイルにエクスポートして **pk12util** を使用してインポートして、ブラウザーの証明書およびキーデータベースを **.redhat-idm-console** ディレクトリーにコピーします。(この手順では、証明書がブラウザーから **.p12** ファイルにエクスポートされていることを前提とします)。

1. ブラウザーから **admin.p12** などのファイルに管理者ユーザー証明書および鍵をエクスポートします。
2. ユーザーのコンソールディレクトリーを開きます。

```
/user-directory/.redhat-idm-console
```

3. 必要な場合は、新規セキュリティーデータベースを作成します。

```
certutil -N -d .
```

4. Certificate System インスタンスを停止します。

```
pki-server stop instance_name
```

5. **pk12util** を使用して証明書をインポートします。

```
# pk12util -i /tmp/admin.p12 -d /user-directory/.redhat-idm-console -W [p12filepassword]
```

手順に成功すると、コマンドは以下を出力します。

```
pk12util: PKCS12 IMPORT SUCCESSFUL
```

6. ブラウザーから発行される CA 証明書の 64 ビットブロブをエクスポートし、これを **ca.crt** ファイルなどに保存します。

7. 管理者ユーザー証明書に関連するベース 64-blob から CA 証明書をインポートします。

```
certutil -A -d . -n ca -t CT,C,C -i ./ca.crt
```

8. Certificate System インスタンスを起動します。

```
pki-server start instance_name
```

9. コンソールを起動します。これで、証明書の入力を求められます。

14.4. JAVA SECURITY MANAGER でのサブシステムの実行

Java サービスには、アプリケーションを実行するための安全でない操作と安全な操作を定義する Security Manager オプションがあります。サブシステムがインストールされると、Security Manager が自動的に有効になります。つまり、各 Tomcat インスタンスは Security Manager が実行されている状態で起動します。

14.4.1. Security Manager ポリシーファイルについて

5 つの Java サブシステム (CA、OCSP、KRA、TKS、および TPS) が Java Security Manager 内で実行されると、以下の 3 つのポリシーの組み合わせを使用します。

- **/usr/share/tomcat/conf** ディレクトリーにあるデフォルトの Tomcat ポリシーからの **catalina.policy** ファイル。これは、一般的な Tomcat ファイルが更新されるたびに更新されます。
- サブシステムインスタンスで提供される **/var/lib/pki/*instance_name*/*subsystem_type*/conf** ディレクトリー内の **pki.policy** ファイル。
- ユーザー定義のセキュリティーポリシーを含む **/var/lib/pki/*instance_name*/*subsystem_type*/conf** ディレクトリーの **custom.policy** ファイル。

この 3 つのファイルは、Tomcat サービスが修正した **catalina.policy** ファイルの作成を開始すると常に連結されます。また、インスタンスに使用される **/var/lib/pki/*instance_name*/*subsystem_type*/conf** ディレクトリーにもこの 3 つのファイルが連結されます。

デフォルトの **pki.policy** ファイルには、PKI サブシステムが使用する Tomcat、LDAP、およびシンボリックリンクサービスへの無制限のアクセスを付与するパーミッションが含まれます。以下に例を示します。

```
// These permissions apply to Tomcat java as utilized by PKI instances
grant codeBase "file:/usr/share/java/tomcat/-" {
    permission java.security.AllPermission;
};
```

この **custom.policy** ファイルはデフォルトでは空になっています。管理者は、指定の PKI ポリシーおよび Tomcat ポリシーに加えて、このファイルでポリシーを作成することができます。

14.4.2. Java Security Manager を使用しないサブシステムインスタンスの起動

PKI Tomcat インスタンスで設定されたすべての Java サブシステムは、Java Security Manager で自動的に実行されます (インスタンスが、**/etc/pki/default.cfg** ファイルの [Tomcat] セクションの下の **pki_security_manager=true** をオーバーライドすることによって作成された場合を除く)。ただし、以

下のように、インスタンスを起動または再起動して、Java Security Manager を起動 **せず** に実行することができます。

手順14.1 Java Security Manager を使用しないインスタンスの起動

1. インスタンスを停止します。

```
# pki-server stop instance_name
```

2. **/etc/sysconfig/*instance_name*** ファイルを編集し、セキュリティーマネージャーをオフにします。

```
SECURITY_MANAGER="false"
```

3. インスタンスを起動します。

```
# pki-server start instance_name
```

14.5. LDAP データベースの設定

Certificate System は、受信した要求に応じて証明書管理機能およびキー管理機能を実行します。これらの機能には、以下が含まれます。

- 証明書要求の保存および取得
- 証明書レコードの保存および取得
- CRL の保存
- ACL の保存
- 特権ユーザーとロール情報の保存
- エンドユーザーの暗号化秘密鍵レコードの保存および取得

これらの機能を満たすために、Certificate System は *内部データベース* または *ローカルデータベース* と呼ばれる Red Hat Directory Server に組み込まれています。Directory Server は、Certificate System 設定の一部として参照されます。Certificate System サブシステムが設定されている場合は、新しいデータベースが Directory Server 内に作成されます。このデータベースは、Certificate System インスタンスのみが組み込みデータベースとして使用し、Directory Server に同梱されているディレクトリー管理ツールを使用して管理できます。

Certificate System インスタンスデータベースは、**serverRoot/slapd-DS_name/db/** 内の他のディレクトリーサーバーデータベースと一緒に一覧表示されます。これらのデータベースは、**/etc/pki/default.cfg** ファイル内の指定されたサブシステムセクションの下の変数 **pki_ds_database** の値によって決定される値によって名前が付けられます (デフォルトでは **CS_instance_name-CA**、**CS_instance_name-KRA**、**CS_instance_name-OCSP**、**CS_instance_name-TKS**、および **CS_instance_name-TPS**)。これは、インスタンス設定時に指定されるデフォルトの形式です。たとえば、証明書マネージャーが **ca1** の場合、データベース名は **ca1-CA** になります。同様に、データベース名は、**/etc/pki/default.cfg** ファイル内の指定されたサブシステムセクション下の **pki_ds_base_dn** の値によって決定されます (デフォルトでは o=CS_instance_name-CA、o=CS_instance_name-KRA、o=CS_instance_name-OCSP、o=CS_instance_name-TKS、または o=CS_instance_name-TPS)。

サブシステムはデータベースを使用して異なるオブジェクトを保存します。証明書マネージャーはすべてのデータ、証明書要求、証明書、CRL、および関連情報を格納しますが、KRA はキーレコードと関連データのみを格納します。

警告

内部データベーススキーマは、Certificate System データのみを格納するよう設定されます。変更を加えたり、他の LDAP ディレクトリーを使用するように Certificate System を設定したりしないでください。これを行うと、データが失われる場合があります。

また、他の目的では内部 LDAP データベースを使用しないでください。

14.5.1. 内部データベース設定の変更

サブシステムインスタンスが内部データベースとして使用する Directory Server インスタンスを変更するには、以下を実行します。

1. サブシステム管理コンソールにログインします。

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

2. **Configuration** タブで **Internal Database** タブを選択します。

3. ホスト名、ポート、およびバインド DN フィールドを変更して、Directory Server インスタンスを変更します。

   ホスト名は、Directory Server がインストールされているマシンの完全修飾ホスト名です (**certificates.example.com** など)。Certificate System はこの名前を使用してディレクトリーにアクセスします。

   デフォルトでは、内部データベースとして使用されている Directory Server インスタンスのホスト名は、実際のホスト名ではなく、**localhost** と表示されます。これは、**localhost** のサーバーはローカルマシンからしかアクセスできないため、内部データベースがシステムの外部に表示されないようにするために行われます。したがって、デフォルト設定では、ローカルマシンからこの Directory Server インスタンスに接続する責任が最小限に抑えられます。

   ホスト名は、内部データベースの可視性がローカルサブネットに制限されている場合は、**localhost** 以外のものに変更できます。たとえば、Certificate System と Directory Server が負荷分散のために別のマシンにインストールされている場合は、Directory Server がインストールされているマシンのホスト名を指定します。

   ポート番号は、Directory Server と SSL 以外の通信に使用される TCP/IP ポートです。

   DN は Directory Manager DN である必要があります。Certificate System サブシステムは、ディレクトリーツリーにアクセスしてディレクトリーと通信する際にこの DN を使用します。

4. **Save** をクリックします。

   設定が変更されます。変更にサーバーの再起動が必要な場合は、プロンプトとそのメッセージが表示されます。その場合には、サーバーを再起動します。

注記

pkiconsole が非推奨になりました。

14.5.2. Directory Server で証明書システムが発行する証明書の使用

証明書システムのインストール時に Directory Server への暗号化された接続を使用するには、外部の認証局 (CA) によって発行された証明書または自己署名証明書のいずれかを使用する必要がありました。ただし、証明書システム CA を設定した後、管理者は、証明書システムが発行した証明書に置き換えることがよくあります。

Directory Server が使用する TLS 証明書を、Certificate System が発行した証明書に置き換えるには、次のコマンドを実行します。

1. Directory Server ホストで以下を行います。

   a. Directory Server インスタンスを停止します。

   ```
   # systemctl stop dirsrv@instance_name
   ```

   b. Certificate Signing Request (CSR) を生成します。

   たとえば、2048 ビット RSA 暗号化を使用し、これを ~/**ds.csr** ファイルに保存する CSR を生成するには、以下を実行します。

   ```
   # PKCS10Client -d /etc/dirsrv/slapd-instance_name/ -p password -a rsa -l 2048 -o
   ~/ds.csr -n "CN=$HOSTNAME"
   PKCS10Client: Debug: got token.
   PKCS10Client: Debug: thread token set.
   PKCS10Client: token Internal Key Storage Token logged in...
   PKCS10Client: key pair generated.
   PKCS10Client: CertificationRequest created.
   PKCS10Client: b64encode completes.
   Keypair private key id: -3387b397ebe254b91c5d6c06dc36618d2ea8b7e6

   -----BEGIN CERTIFICATE REQUEST-----
   ...
   -----END CERTIFICATE REQUEST-----
   PKCS10Client: done. Request written to file: ~/ds.csr
   ```

   c. Directory Server インスタンスを起動し、CA が要求の処理できるようにします。

   ```
   # systemctl start dirsrv@instance_name
   ```

   d. CSR を Certificate System の CA に送信します。以下に例を示します。

   ```
   # pki -d /etc/dirsrv/slapd-instance_name/ ca-cert-request-submit --profile caServerCert --
   csr-file ~/ds.csr
   ----------------------------
   Submitted certificate request
   ----------------------------
     Request ID: 13
   ```

```
Type: enrollment
Request Status: pending
Operation Result: success
```

2. Certificate System ホストで以下を行います。

   a. CA エージェント証明書を Network Security Services (NSS) データベースにインポートして、CMC フル要求を署名します。

      i. 新しいディレクトリーを作成します。以下に例を示します。

      ```
      # mkdir ~/certs_db/
      ```

      ii. 新たに作成したディレクトリーでデータベースを初期化します。

      ```
      # certutil -N -d ~/certs_db/
      ```

      iii. CA 署名証明書のシリアル番号を表示します。

      ```
      # pki -p 8080 ca-cert-find --name "CA Signing Certificate"
      ---------------
      1 entries found
      ---------------
      Serial Number: 0x87bbe2d
      ...
      ```

      iv. 前の手順のシリアル番号を使用して、CA 署名証明書を **~/certs_db/CA.pem** ファイルにダウンロードします。

      ```
      # pki -p 8080 ca-cert-show 0x87bbe2d --output ~/certs_db/CA.pem
      ```

      v. CA 署名証明書を NSS データベースにインポートします。

      ```
      # pki -d ~/certs_db/ -c password client-cert-import "CA Certificate" --ca-cert
      ~/certs_db/CA.pem
      ```

      vi. エージェントの証明書をインポートします。

      ```
      # pk12util -d ~/certs_db/ -i ~/.dogtag/instance_name/ca_admin_cert.p12
      Enter Password or Pin for "NSS FIPS 140-2 Certificate DB": password
      Enter password for PKCS12 file: password
      pk12util: PKCS12 IMPORT SUCCESSFUL
      ```

   b. CMS (CMC) 要求で Certificate Management を作成します。

      i. **~/sslserver-cmc-request.cfg** などの設定ファイルを以下の内容で作成します。

      ```
      # NSS database directory where the CA agent certificate is stored.
      dbdir=~/certs_db/

      # NSS database password.
      password=password
      ```

```
# Token name (default is internal).
tokenname=internal

# Nickname for CA agent certificate.
nickname=caadmin

# Request format: pkcs10 or crmf.
format=pkcs10

# Total number of PKCS10/CRMF requests.
numRequests=1

# Path to the PKCS10/CRMF request.
# The content must be in Base-64 encoded format.
# Multiple files are supported. They must be separated by space.
input=~/ds.csr

# Path for the CMC request.
output=~/sslserver-cmc-request.bin
```

ii. CMC 要求を作成します。

```
# CMCRequest ~/sslserver-cmc-request.cfg
...
The CMC enrollment request in base-64 encoded format:
...
The CMC enrollment request in binary format is stored in ~/sslserver-cmc-request.bin
```

c. CMC 要求を送信します。

i. ~/**sslserver-cmc-submit.cfg** などの設定ファイルを以下の内容で作成します。

```
# PKI server host name.
host=server.example.com

# PKI server port number.
port=8443

# Use secure connection.
secure=true

# Use client authentication.
clientmode=true

# NSS database directory where the CA agent certificate is stored.
dbdir=~/certs_db/

# NSS database password.
password=password

# Token name (default: internal).
tokenname=internal
```

```
# Nickname of CA agent certificate.
nickname=caadmin

# CMC servlet path
servlet=/ca/ee/ca/profileSubmitCMCFull?profileId=caCMCserverCert

# Path for the CMC request.
input=~/sslserver-cmc-request.bin

# Path for the CMC response.
output=~/sslserver-cmc-response.bin
```

ii. 要求を送信します。

```
# HttpClient sslserver-cmc-submit.cfg
...
The response in binary format is stored in
~/sslserver-cmc-response.bin
```

iii. 必要に応じて、結果を確認します。

```
# CMCResponse -d ~/certs_db/ -i ~/sslserver-cmc-response.bin
...
Number of controls is 1
Control #0: CMCStatusInfoV2
  OID: {1 3 6 1 5 5 7 7 25}
  BodyList: 1
  Status: SUCCESS
```

d. Directory Server 証明書のシリアル番号を表示します。

```
# pki -p 8080 ca-cert-find --name "DS Certificate"
---------------
1 entries found
---------------
Serial Number: 0xc3eeb0c
...
```

e. 前の手順でシリアル番号を使用して、証明書をダウンロードします。

```
# pki -p 8080 ca-cert-show 0xc3eeb0c --output ~/ds.crt
```

f. Directory Server および CA 証明書の証明書を Directory Server ホストにコピーします。以下に例を示します。

```
# scp ~/ds.crt ~/certs_db/CA.pem ds.example.com:~/
```

g. Certificate System を停止します。

```
# pki-server stop instance_name
```

3. Directory Server ホストで以下を行います。

a. Directory Server インスタンスを停止します。

```
# systemctl stop dirsrv@instance_name
```

b. 証明書を置き換えます。詳細は、『Red Hat Directory Server 管理ガイド』 の該当するセクションを参照してください。

i. 古い証明書および CA 証明書を削除します。『証明書の削除』 を参照してください。

ii. Certificate System が発行する CA 証明書をインストールします。『認証局証明書のインストール』 を参照してください。

iii. Certificate System が発行する Directory Server の証明書をインストールします。『サーバー証明書のインストール』 を参照してください。

c. Directory Server インスタンスを起動します。

```
# systemctl start dirsrv@instance_name
```

4. Certificate System を開始します。

```
# pki-server stop instance_name
```

5. 必要に応じて、証明書ベースの認証を設定します。詳細は、「内部データベースでの SSL/TLS クライアント認証の有効化」 を参照してください。

14.5.3. 内部データベースでの SSL/TLS クライアント認証の有効化

クライアント認証 により、あるエンティティーが証明書を提示して別のエンティティーに対して認証できるようにします。この認証方法は、たとえば、Certificate System エージェントがエージェントサービスページにログインするために使用します。

Certificate System インスタンスと、内部データベースとして使用する LDAP ディレクトリーインスタンスとの間で SSL/TLS 接続を使用するには、Certificate System インスタンスが LDAP ディレクトリーを認証してバインドできるようにクライアント認証を有効にする必要があります。

クライアント認証の設定には 2 つの部分があります。1 つ目は、SSL/TLS を設定し、Certificate System インスタンスのアクセスを制御するように ACI を設定するなど、LDAP ディレクトリーを設定します。2 つ目は、LDAP ディレクトリーにバインドして、証明書を設定するのに使用する Certificate System インスタンスでユーザーを作成します。

PKI インスタンスの LDAPS を設定するには、pkispawn(8) man ページの (例: セキュアな LDAP 接続を使用した PKI サブシステムのインストール) を参照してください。

14.5.4. 内部データベースへのアクセス制限

Red Hat Directory Server コンソールは、Certificate System が内部データベースとして使用する Directory Server インスタンスのエントリーまたはアイコンを表示します。

Certificate System 管理者権限を持つユーザーにアクセスが制限されている Certificate System Console とは異なり、Directory Server Console は、どのユーザーからでもアクセスすることができる。ユーザーは、内部データベースの Directory Server Console を開いて、そこに保存されているデータに変更できます (Certificate System 管理者グループからユーザーを削除したり、グループに自分のエントリーを追加したりなど)。

アクセスは、Directory Manager DN とパスワードを知っているユーザーのみに内部データベースに制限できます。このパスワードは、シングルサインオンのパスワードキャッシュを変更することで変更できます。

1. Directory Server コンソールにログインします。
2. Certificate System 内部データベースエントリーを選択して、**Open** をクリックします。
3. **Configuration** タブを選択します。
4. ナビゲーションツリーで **Plug-ins** を展開し、**Pass-Through Authentication** を選択します。
5. 右側のペインで、**Enable plugin** チェックボックスの選択を解除します。
6. **Save** をクリックします。

   サーバーを再起動するように要求します。
7. **Tasks** タブをクリックして、**Restart the Directory Server** をクリックします。
8. Directory Server コンソールを閉じます。
9. サーバーが再起動したら、内部データベースインスタンスの Directory Server コンソールを開きます。

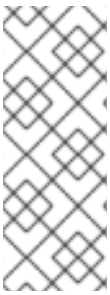
   Login to Directory ダイアログボックスが表示されます。**Distinguished Name** フィールドが Directory Manager DN を表示し、パスワードを入力します。

   内部データベースの Directory Server Console は、正しいパスワードを入力する場合のみ開きます。

14.6. セキュリティードメイン設定の表示

セキュリティードメイン は PKI サービスのレジストリーです。CA などの PKI サービスは、これらのドメインに独自の情報を登録するため、PKI サービスのユーザーはレジストリーを確認して他のサービスを検索できます。Certificate System のセキュリティードメインサービスは、Certificate System サブシステムの PKI サービスの登録と、共有信頼ポリシーのセットの両方を管理します。

セキュリティードメインはサブシステム間の信頼関係を自動的に管理するため、TPS、TKS、および KRA が同じセキュリティードメイン内にある場合は安全に通信できます。



注記

セキュリティードメインはサブシステムの設定時に使用されます。サブシステムが設定されていると、セキュリティードメインレジストリーを確認して利用可能なインスタンスを確認することができます。TKS と KRA を操作に使用する TPS など、別のインスタンスとの信頼関係を作成する必要がある場合は、セキュリティードメインを使用して、選択した TKS インスタンスと KRA インスタンスに TPS エージェントユーザーを作成します。

レジストリーは、ドメイン内でサブシステムによって提供されるすべての PKI サービスの完全なビューを提供します。各 Certificate System サブシステムは、ホストまたはセキュリティードメインのメンバーのいずれかである必要があります。

CA のみがセキュリティードメインをホストおよび管理できます。各 CA には独自の LDAP エントリーがあり、セキュリティードメインは CA エントリーの下にある組織グループです。

```
ou=Security Domain,dc=example,dc=com
```

次に、セキュリティードメイン組織グループの下に各サブシステムタイプのリストがあり、グループタイプを識別するための特別なオブジェクトクラス (**pkiSecurityGroup**) があります。

```
cn=KRAList,ou=Security Domain,dc=example,dc=com
objectClass: top
objectClass: pkiSecurityGroup
cn: KRAList
```

各サブシステムインスタンスは、エントリータイプを識別するための特別な **pkiSubsystem** オブジェクトクラスを使用してそのグループのメンバーとして保存されます。

```
dn: cn=server.example.com:8443,cn=KRAList,ou=Security Domain,dc=example,dc=com
objectClass: top
objectClass: pkiSubsystem
cn: kra.example.com:8443
host: server.example.com
SecurePort: 8443
SecureAgentPort: 8443
SecureAdminPort: 8443
UnSecurePort: 8080
DomainManager: false
Clone: false
SubsystemName: KRA server.example.com 8443
```

14.7. サブシステムの SELINUX ポリシーの管理

SELinux は、承認されていないアクセスと改ざんを制限するためにシステム全体で実施される、必須のアクセス制御ルールのコレクションです。SELinux の詳細は、Red Hat Enterprise Linux 8 の 『SELinux の使い方ガイド』 を参照してください。

14.7.1. SELinux について

基本的に、SELinux はシステム上の **オブジェクト** を識別します。これは、ファイル、ディレクトリー、ユーザー、プロセス、ソケット、または Linux ホスト上その他の物になります。これらのオブジェクトは Linux API オブジェクトに対応します。各オブジェクトは **セキュリティーコンテキスト** にマッピングされ、オブジェクトのタイプと、Linux サーバーでの機能が可能になります。

システムプロセスは SELinux ドメイン内で実行されます。各ドメインには、SELinux ドメインがシステム上の他の SELinux オブジェクトと対話する方法を定義する一連のルールがあります。この一連のルールは、プロセスがアクセスできるリソースと、それらのリソースで実行できる操作を決定します。

Certificate System では、各サブシステムタイプはそのサブシステムタイプの特定のドメイン内で実行されます。そのサブシステムタイプのすべてのインスタンスは、システム上のインスタンスの数に関係なく、同じ SELinux ドメインに属します。たとえば、サーバーに 3 つの CA がインストールされている場合は、その 3 つがすべて **http_port_t** SELinux ドメインに属しています。

すべてのサブシステムのルールと定義は、Certificate System SELinux ポリシー全体を設定します。Certificate System SELinux ポリシーは、サブシステムのインストール時に設定され、**pkispawn** でサブシステムが追加されたり、**pkidestroy** で削除されるたびに、すべての SELinux ポリシーが更新されます。

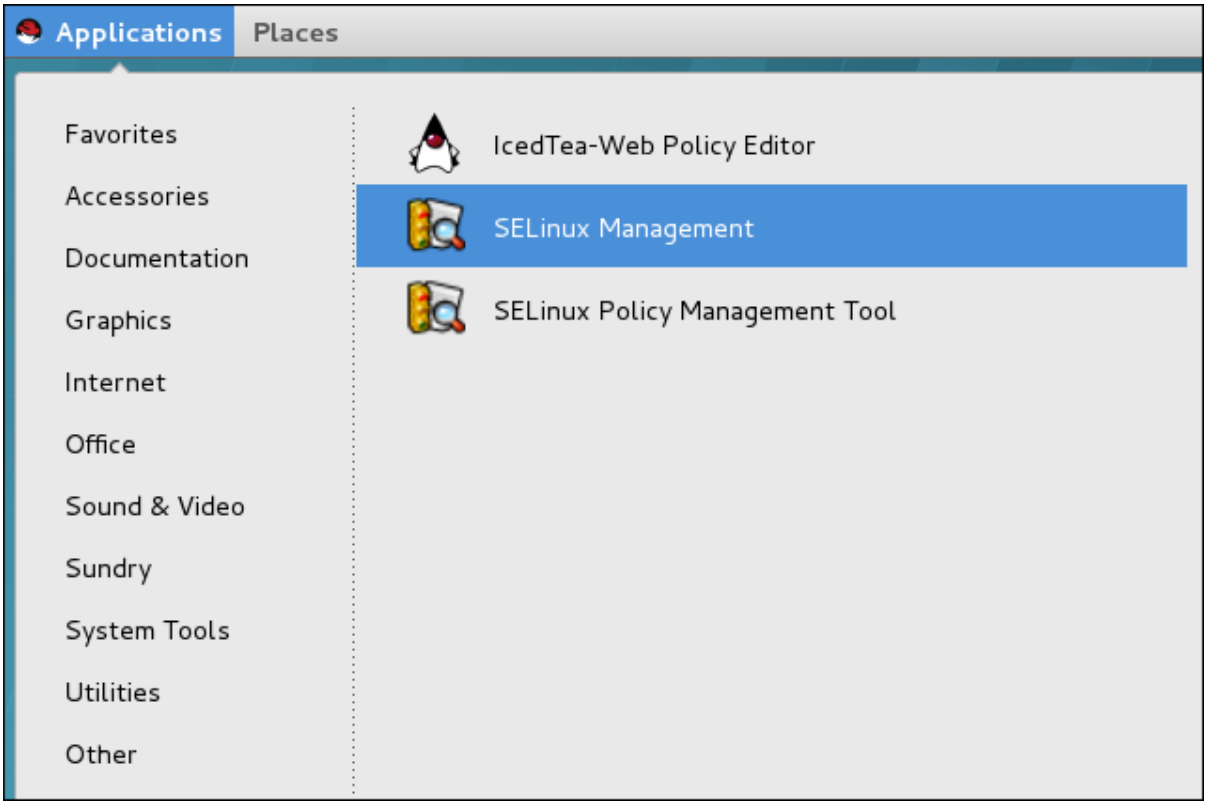
Certificate System サブシステムは、SELinux を強制モードに設定して実行します。つまり、すべての SELinux ルールに従う必要がある場合でも、Certificate System の操作を正常に実行できます。

デフォルトでは、Certificate System サブシステムは SELinux ポリシーにより制限が限定されます。

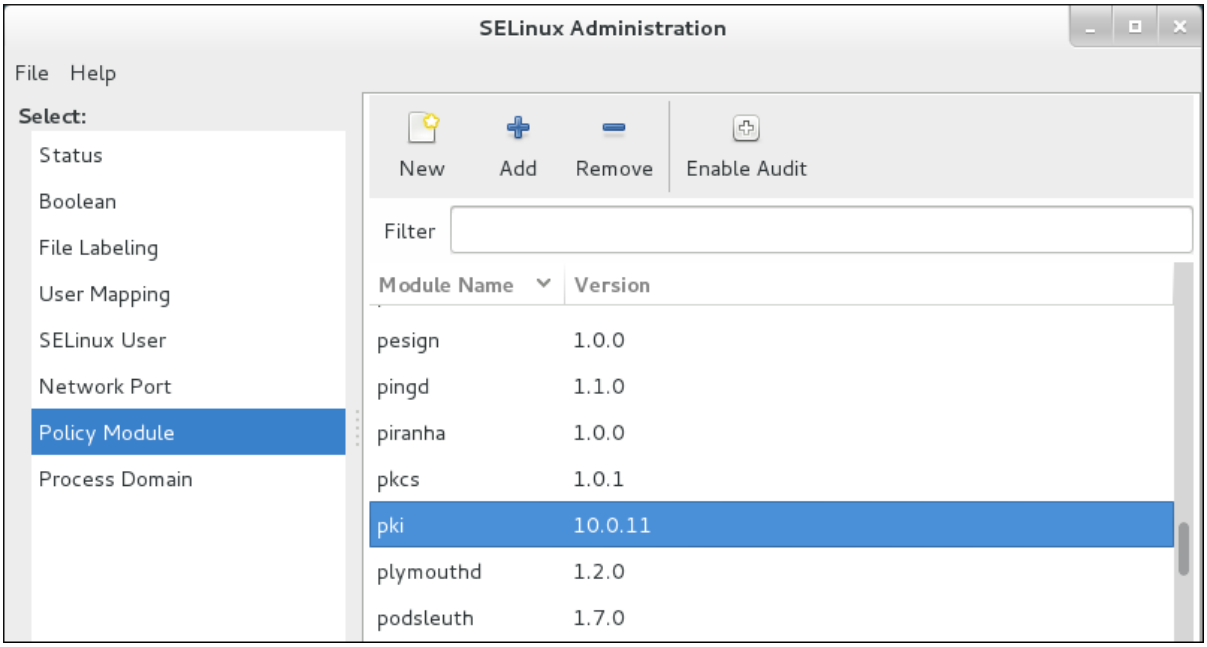
14.7.2. サブシステムの SELinux ポリシーの表示

すべての Certificate System ポリシーは、システムの SELinux ポリシーに含まれます。設定したポリシーは、SELinux 管理 GUI を使用して表示できます。この GUI は、policycoreutils-gui パッケージをインストールできます。

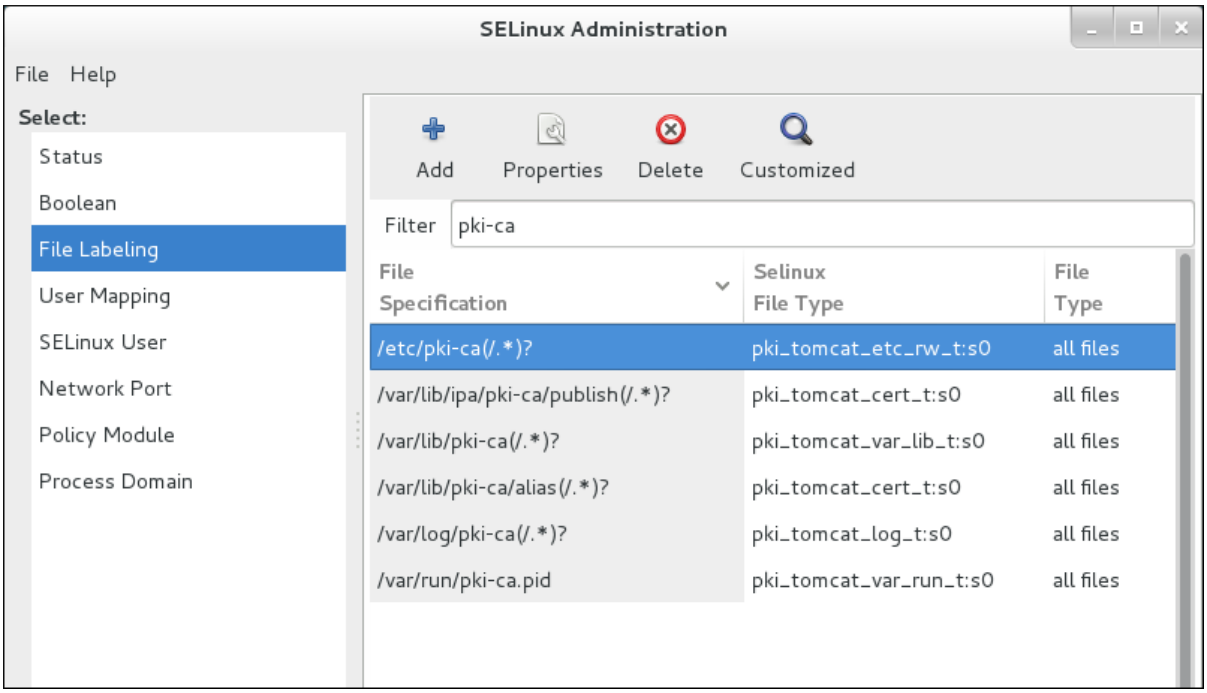
1. **system-config-selinux** コマンドを実行するか、メインのシステムメニュー用の **Applications → Other → SELinux Management** にアクセスしてユーティリティーを開きます。



2. インストールされている Certificate System SELinux ポリシーのバージョンを確認するには、左側のバーの **Policy Module** セクションを参照してください。



3. 個々のファイルおよびプロセスに設定したポリシーを表示するには、**File Labeling** セクションをクリックします。サブシステムのポート割り当てのポリシーを表示するには、**Network Port** セクションをクリックします。



14.7.3. nCipher netHSM コンテキストの再ラベル付け

nCipher netHSM ソフトウェアには独自の SELinux ポリシーが付属していないため、Certificate System には、例14.1「netHSM SELinux ポリシー」 に示すデフォルトの netHSM ポリシーが含まれています。

例14.1 netHSM SELinux ポリシー

```
# default labeling for nCipher
/opt/nfast/scripts/init.d/(.*)  gen_context(system_u:object_r:initrc_exec_t,s0)
/opt/nfast/sbin/init.d-ncipher  gen_context(system_u:object_r:initrc_exec_t,s0)
/opt/nfast(/.*)?                gen_context(system_u:object_r:pki_common_t, s0)
/dev/nfast(/.*)?                gen_context(system_u:object_r:pki_common_dev_t,0)
```

他のルールを使用すると、**pki_*_t** ドメインが **pki_common_t** と **pki_common_dev_t** のラベルが付いたファイルと通信できます。

(デフォルトディレクトリー **/opt/nfast** であっても) nCipher 設定のいずれかを変更した場合は、**restorecon** を実行して、ファイルがすべて適切にラベル付けされていることを確認します。

```
restorecon -R /dev/nfast
restorecon -R /opt/nfast
```

nCipher ソフトウェアが別の場所にインストールされている場合や、HSM が異なる場合は、**semanage** を使用してデフォルトの Certificate System HSM ポリシーを再ラベル付けする必要があります。

14.8. 証明書システムのバックアップと復元

Certificate System には、バックアップと復元のツールは含まれません。ただし、Certificate System コンポーネントは手動でアーカイブおよび復元できます。このコンポーネントは、証明書または鍵の情報が失われた場合に情報にアクセスできないデプロイメントに必要な場合があります。
Certificate System の主な部分は、データ損失やハードウェアに障害が発生した場合に、通常 3 つにバックアップする必要があります。

- **内部データベース。**サブシステムは LDAP データベースを使用してデータを保管します。Directory Server は、独自のバックアップスクリプトと手順を提供します。
- **セキュリティーデータベース。**セキュリティーデータベースは、証明書とキー情報を保管します。これらが HSM に保存されている場合は、データをバックアップする方法は HSM ベンダーのドキュメントを参照してください。情報がインスタンスの **alias** ディレクトリーのデフォルトディレクトリーに保存されている場合は、インスタンスディレクトリーを使用してバックアップします。これを個別に作成するには、**tar** または **zip** などのユーティリティーを使用します。
- **インスタンスディレクトリー。**インスタンスディレクトリーには、すべての設定ファイル、セキュリティーデータベース、その他のインスタンスファイルが含まれます。これは、**tar** や **zip** などのユーティリティーを使用してバックアップできます。

14.8.1. LDAP 内部データベースのバックアップおよび復元

Red Hat Directory Server のドキュメント には、データベースのバックアップおよび復元の詳細情報が記載されています。

14.8.1.1. LDAP 内部データベースのバックアップ

Directory Server インスタンスのバックアップには、**dsctl** コマンドのサブコマンドのペアが 2 つ利用できます。それぞれのバックアップサブコマンドには、生成したファイルを復元するためのカウンターがあります。

- **db2ldif** サブコマンドは、**ldif2db** サブコマンドを使用して復元できる LDIF ファイルを作成します。
- **db2bak** サブコマンドは、**bak2db** サブコマンドを使用して復元できるバックアップファイルを作成します。

14.8.1.1.1. db2ldif を使用したバックアップ

db2ldif サブコマンドを実行すると、単一のサブシステムデータベースのバックアップが作成されます。

注記

db2ldif サブコマンドは **dirsrv** ユーザーで実行するため、**/root/** ディレクトリー配下に書き込み権限がないので、書き込みが可能なパスを提供する必要があります。

PKI サブシステムが使用する各 Directory Server データベースをバックアップします。以下の **pki-server ca-db-config-show** コマンドを使用して、指定のサブシステムのデータベース名を確認することができます。たとえば、メインデータベースのバックアップを作成するには、**userRoot** を使用します。

1. インスタンスを停止します。

```
# dsctl instance_name stop
```

2. データベースを LDIF ファイルにエクスポートします。

```
# dsctl instance_name db2ldif userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
ldiffile: /tmp/example.ldif
[18/Jul/2018:10:46:03.353656777 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
[18/Jul/2018:10:46:03.383101305 +0200] - INFO - ldbm_back_ldbm2ldif - export userroot:
Processed 160 entries (100%).
[18/Jul/2018:10:46:03.391553963 +0200] - INFO - dblayer_pre_close - All database threads
now stopped
db2ldif successful
```

3. インスタンスを起動します。

```
# dsctl instance_name start
```

ldif2db サブコマンドを使用して LDIF ファイルを復元するには、「ldif2db を使用した復元」 を参照してください。

14.8.1.1.2. db2bak を使用したバックアップ

db2bak サブコマンドを実行すると、Directory Server (および Directory Server インスタンスが維持するその他のデータベース) の全 Certificate System サブシステムデータベースがバックアップされます。以下に例を示します。

以下に例を示します。

1. インスタンスを停止します。

```
# dsctl instance_name stop
```

2. データベースのバックアップを作成します。

```
# dsctl instance_name db2bak
```

```
OK group dirsrv exists
OK user dirsrv exists
[18/Jul/2018:14:02:37.358958713 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
...
db2bak successful
```

3. インスタンスを起動します。

```
# dsctl instance_name start
```

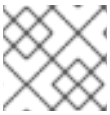
注記

db2bak サブコマンドを **dirsrv** ユーザーで実行するため、ターゲットディレクトリーは **dirsrv** で書き込み可能でなければなりません。引数を指定せずにコマンドを実行すると、**db2bak** が適切な書き込み権限を持つ **/var/lib/dirsrv/slapd-*<instance_name>*/bak** ディレクトリーにバックアップが作成されます。

bak2db を使用して LDIF ファイルを復元するには、「bak2db を使用した復元」 を参照してください。

14.8.1.2. LDAP 内部データベースの復元

Directory Server インスタンスをバックアップする方法に応じて、対応するファイルで **ldif2db**または**bak2db** を使用して、データベースを復元してください。



注記

データベースを復元する前に、インスタンスを停止してください。

14.8.1.2.1. ldif2db を使用した復元

db2ldif で LDIF ファイルを作成している場合は、**ldif2db** サブコマンドを使用して、Directory Server インスタンスを停止してファイルをインポートします。1つのデータベースを指定して、バックアップから復元できます。たとえば、メインデータベースの場合、**userRoot** などです。

1. Directory Server インスタンスを停止します。

```
# dsctl instance_name stop
```

2. LDIF ファイルからデータをインポートします。

```
# dsctl instance_name ldif2db userroot /tmp/example.ldif
OK group dirsrv exists
OK user dirsrv exists
[17/Jul/2018:13:42:42.015554231 +0200] - INFO - ldbm_instance_config_cachememsize_set
- force a minimal value 512000
...
[17/Jul/2018:13:42:44.302630629 +0200] - INFO - import_main_offline - import userroot:
Import complete.  Processed 160 entries in 2 seconds. (80.00 entries/sec)
ldif2db successful
```

3. Directory Server インスタンスを開始します。

```
# dsctl instance_name start
```

14.8.1.2.2. bak2db を使用した復元

db2bak でバックアップファイルを作成した場合は、Directory Server を停止し、**bak2db** サブコマンドを使用してファイルをインポートします。以下に例を示します。

1. Directory Server インスタンスを停止します。

```
# dsctl instance_name stop
```

2. データベースを復元します。

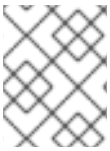
```
# dsctl instance_name bak2db /var/lib/dirsrv/slapd-instance_name/bak/instance_name-time_stamp/
OK group dirsrv exists
OK user dirsrv exists
[20/Jul/2018:15:52:24.932598675 +0200] - INFO - ldbm_instance_config_cachememsize_set - force a minimal value 512000
...
bak2db successful
```

3. Directory Server インスタンスを開始します。

```
# dsctl instance_name start
```

14.8.2. インスタンスディレクトリーのバックアップおよび復元

インスタンスディレクトリーには、サブシステムインスタンスのすべての設定情報が含まれているため、インスタンスディレクトリーのバックアップを作成すると、内部データベースに含まれていない設定情報が保持されます。



注記

インスタンスまたはセキュリティーデータベースをバックアップする前にサブシステムインスタンスを停止します。

1. サブシステムインスタンスを停止します。

```
pki-server stop instance_name
```

2. ディレクトリーを圧縮ファイルに保存します。

```
# cd /var/lib/pki/
# tar -chvf /export/archives/pki/instance_name.tar instance_name/
```

以下に例を示します。

```
# cd /var/lib/pki/
# tar -chvf /tmp/test.tar pki-tomcat/ca/
pki-tomcat/ca/
```

```
pki-tomcat/ca/registry/
pki-tomcat/ca/registry/ca/
.....
```

3. サブシステムインスタンスを再起動します。

```
pki-server start instance_name
```

データが破損したり、ハードウェアが破損している場合に、Certificate System のバックアップファイル (**alias** バックアップおよび完全なインスタンスのディレクトリーバックアップ) の両方を使用して、現在のディレクトリーを交換できます。データを復元するには、**unzip** ツールまたは **tar** ツールを使用してアーカイブファイルを圧縮解除し、既存のファイルにアーカイブをコピーします。

インスタンスディレクトリーを復元するには、以下を実行します。

1. アーカイブを展開します。

```
cd /export/archives/pki/
tar -xvf instance_name.tar
```

以下に例を示します。

```
# cd /tmp/
# tar -xvf test.tar
pki-tomcat/ca/
pki-tomcat/ca/registry/
pki-tomcat/ca/registry/ca/
pki-tomcat/ca/registry/ca/default.cfg
.....
```

2. サブシステムインスタンスが停止していない場合は停止します。

```
pki-server stop instance_name
```

3. アーカイブされたファイルをコピーして、インスタンスディレクトリーを復元します。

```
cp -r /export/archives/pki/instance_name /var/lib/pki/instance_name
```

以下に例を示します。

```
# cp -r /tmp/pki-tomcat/ca/ /var/lib/pki/pki-tomcat/ca/
```

4. 復元されたファイルの所有権およびグループのパーミッションが **pkiuser** に設定されていることを確認します。

```
# chown -R pkiuser:pkiuser /var/lib/pki/pki-tomcat/ca/
```

5. サブシステムインスタンスを再起動します。

```
pki-server start instance_name
```

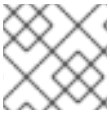
14.9. セルフテストの実行

Certificate System には、サーバーのセルフテストを可能にする機能が追加されました。セルフテストは起動時に実行され、オンデマンドで実行することもできます。起動セルフテストはサーバーの起動時に実行され、重要なセルフテストが失敗した場合にサーバーが起動しないようにします。オンデマンドのセルフテストは、サブシステムコンソールのセルフテストボタンをクリックして実行されます。

14.9.1. セルフテストの実行

CA、OCSP、KRA、または TKS サブシステムのオンデマンドのセルフテストは、コンソールから実行します。TPS システムのオンデマンドのセルフテストは、Web サービスページから実行されます。

14.9.1.1. コンソールからのセルフテストの実行



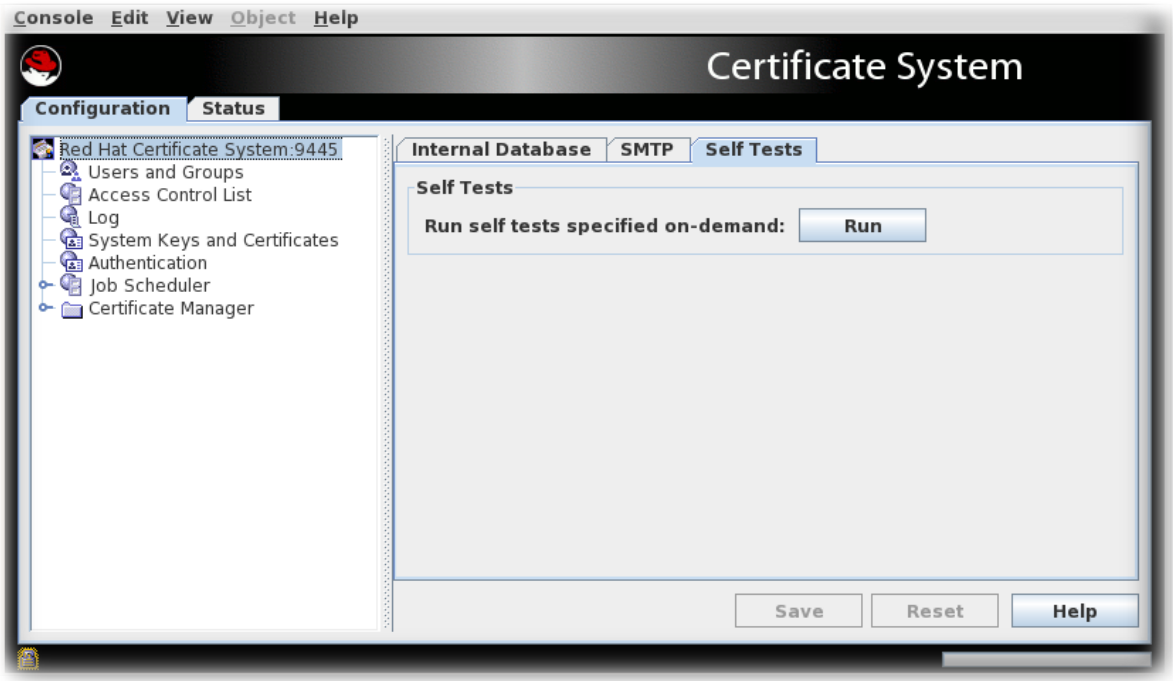
注記

pkiconsole が非推奨になりました。

1. コンソールにログインします。

```
pkiconsole https://server.example.com:admin_port/subsystem_type
```

2. 左側のペインの上部にあるサブシステム名を選択します。



3. **Self Tests** タブを選択します。

4. **Run** をクリックします。

   サブシステムに設定されたセルフテストが実行されます。重大なセルフテストに失敗すると、サーバーが停止します。

5. **On-Demand Self Tests Results** ウインドウが表示され、セルフテストの実行にログイベントが表示されます。

14.9.1.2. TPS セルフテストの実行

コマンドラインインターフェイス (CLI) から TPS のセルフテストを実行するには、以下を実行します。

- **pki tps-selftest-find**
- **pki tps-selftest-run**
- **pki tps-selftest-show**

14.9.2. セルフテストロギング

別のログ (**selftest.log**) が、起動用セルフテストとオンデマンドセルフテストの両方のレポートが含まれるログディレクトリーに追加されます。このログは、**CS.cfg** ファイルのログの設定を変更することで設定されます。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『セルフテスト設定の修正』 セクションを参照してください。

14.9.3. POSIX システム ACL の設定

POSIX システムアクセス制御ルールは、システムユーザーのパーミッションに対してより細かい粒度を提供します。これらの ACL は、インスタンスが完全に設定された後、インスタンスごとに設定する必要があります。ACL の詳細は、Red Hat Enterprise Linux システム管理ガイドの該当する章 を参照してください。

14.9.3.1. CA、KRA、OCSP、TKS、および TPS の POSIX システム ACL の設定

ext4 や XFS などの最新のファイルシステムはデフォルトで ACL を有効にし、最新の Red Hat Enterprise Linux インストールで使用されます。

1. インスタンスを停止します。

```
pki-server stop instance_name
```

2. インスタンスのディレクトリーおよびファイルに対する読み取り可能性を pkiadmin グループに設定します。

```
# setfacl -R -L -m g:pkiadmin:r,d:g:pkiadmin:r /var/lib/pki/instance_name
```

3. すべてのディレクトリーに実行 (x) ACL パーミッションを適用します。

```
# find -L /var/lib/pki/instance_name -type d -exec setfacl -L -n -m
g:pkiadmin:rx,d:g:pkiadmin:rx {} \;
```

4. インスタンスの signedAudit/ ディレクトリーおよびその関連ファイルから、pkiadmin グループのグループの読み取り性を削除します。

```
# setfacl -R -L -x g:pkiadmin,d:g:pkiadmin /var/lib/pki/instance_name/logs/signedAudit
```

5. インスタンスの signedAudit/ ディレクトリーとその関連ファイルの pkiaudit グループのグループの読み取り性を設定します。

```
# setfacl -R -L -m g:pkiaudit:r,d:g:pkiaudit:r /var/lib/pki/instance_name/logs/signedAudit
```

6. signedAudit/ ディレクトリーとそのすべてのサブディレクトリーで実行 (x) ACL パーミッションを再適用します。

```
# find -L /var/lib/pki/instance_name/logs/signedAudit -type d -exec setfacl -L -n -m
g:pkiaudit:rx,d:g:pkiaudit:rx {} \;
```

7. インスタンスを起動します。

```
pki-server start instance_name
```

8. 別のログ (**selftest.log**) が、起動用セルフテストとオンデマンドセルフテストの両方のレポートが含まれるログディレクトリーに追加されます。

```
# getfacl /var/lib/pki/instance_name
/var/lib/pki/instance_name/subsystem_type/logs/signedAudit/
getfacl: Removing leading '/' from absolute path names
# file: var/lib/pki/instance_name
# owner: pkiuser
# group: pkiuser
user::rwx
group::rwx
group:pkiadmin:r-x
mask::rwx
other::r-x
default:user::rwx
default:group::rwx
default:group:pkiadmin:r-x
default:mask::rwx
default:other::r-x

# file: var/lib/pki/instance_name/logs/signedAudit
# owner: pkiuser
# group: pkiaudit
user::rwx
group::rwx
group:pkiaudit:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:pkiaudit:r-x
default:mask::rwx
default:other::---
```

第15章 証明書システムユーザーおよびグループの管理

本章では、管理、エージェントサービス、およびエンドエンティティーページにアクセスするための承認を設定する方法を説明します。

15.1. 承認について

承認 は、Certificate System に関連付けられた特定のタスクにアクセスできるようにするプロセスです。アクセスは、特定のユーザーまたはグループのサブシステムの特定領域に対応し、異なるユーザーおよびグループに対して異なるタスクを許可できるように制限できます。

ユーザーは、作成されるサブシステムに固有のものです。各サブシステムには、インストールされている他のサブシステムから独立した独自のユーザーセットがあります。ユーザーはグループに配置され、事前定義またはユーザーの作成が可能です。**アクセス制御リスト** (ACL) で、権限がグループに割り当てられます。管理コンソール、エージェントサービスインターフェイス、およびエンドエンティティーページの領域に関連付けられた ACL があり、操作の続行を許可する前に許可チェックを実行します。各 ACL の**アクセス制御命令** (ACI) が作成され、その ACL が指定されたユーザー、グループ、または IP アドレスに対して許可される操作を許可または拒否します。

ACL には、作成されるデフォルトグループのデフォルト ACI セットが含まれます。これらの ACI は、事前に定義されたグループの権限を変更するか、新たに作成したグループに権限を割り当てるように変更できます。

承認は以下のプロセスを経て行われます。

1. ユーザーは、Certificate System ユーザー ID とパスワードまたは証明書を使用してインターフェイスに対して認証します。

2. サーバーは、データベースに保存されているユーザー ID とパスワードが一致するか、データベースに保存されているものに対して証明書をチェックして、ユーザーを認証します。証明書ベースの認証では、サーバーは証明書が有効であることも確認し、証明書の DN をユーザーに関連付けてユーザーエントリーを確認することにより、ユーザーのグループメンバーシップを見つけます。パスワードベースの認証では、サーバーはユーザー ID に対してパスワードを確認してから、そのユーザー ID をグループに含まれるユーザー ID に関連付けることにより、ユーザーのグループメンバーシップを検索します。

3. ユーザーが操作を実行しようとすると、承認メカニズムはユーザーのユーザー ID、ユーザーが属するグループ、ユーザーの IP アドレスを、そのユーザー、グループ、または IP アドレスに設定された ACL と比較します。その操作を許可する ACL が存在する場合、操作は続行されます。

15.2. デフォルトグループ

ユーザーの権限は、ユーザーのグループ (ロール) メンバーシップにより決定されます。ユーザーを割り当てることのできる 3 つのグループ (ロール) があります。

- **管理者**。このグループには、管理インターフェイスで利用可能なすべてのタスクへの完全なアクセスが付与されます。

- **エージェント**。このグループには、エージェントサービスインターフェイスで利用可能なすべてのタスクに完全アクセスできます。

- **監査者**。このグループには、署名済み監査ログを表示するためのアクセスが付与されます。このグループには他の権限がありません。

サブシステム間の通信のみに限り作成される 4 つ目のロールがあります。管理者は、実際のユーザーをこのようなロールに割り当てることはできません。

- **エンタープライズ管理者**。各サブシステムインスタンスには、設定中にセキュリティードメインに参加していると、エンタープライズ管理者としてサブシステム固有のロールが自動的に割り当てられます。これらのロールはセキュリティードメインのサブシステム間で信頼できる関係を自動的に提供し、各サブシステムが他のサブシステムと効率的に対話できるようにします。

15.2.1. 管理者

管理者には、すべての管理タスクを実行できるパーミッションがあります。ユーザーは、グループの **Administrators** グループに追加され、管理者として特定されます。このグループの各メンバーには、Certificate System のそのインスタンスに対する管理権限が必要です。

Certificate System インスタンスごとに少なくとも 1 つの管理者を定義する必要がありますが、インスタンスに割り当てることのできる管理者の数に制限はありません。インスタンスの設定時に最初の管理者エントリーが作成されます。

管理者は、Certificate System ユーザー ID とパスワードを使用して単純なバインドで認証されます。

表15.1 セキュリティードメインのユーザーロール

ロール	説明
セキュリティードメインの管理者	• セキュリティードメインのユーザーおよびグループデータベースでユーザーを追加および変更します。 • 共有信頼ポリシーを管理します。 • ドメインサービスのアクセス制御を管理します。 デフォルトでは、ドメインをホストする CA の CA 管理者はセキュリティードメイン管理者として割り当てられます。
エンタープライズ CA 管理者	• ドメインのサブ CA、サーバー、およびサブシステムの証明書を自動的に承認します。 • セキュリティードメインで CA サブシステム情報を登録および登録解除します。
エンタープライズ KRA 管理者	• ドメインの CA からトランスポート、ストレージ、サーバー、およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで KRA サブシステム情報を登録および登録解除します。 • KRA コネクター情報を CA にプッシュします。

ロール	説明
エンタープライズ OCSP 管理者	• ドメイン内の CA から OCSP、サーバー、およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで OCSP サブシステム情報を登録および登録解除します。 • CRL を公開する情報を CA にプッシュします。
エンタープライズ TKS 管理者	• ドメインの CA からサーバー証明書およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで TKS サブシステム情報を登録および登録解除します。
エンタープライズ TPS 管理者	• ドメインの CA からサーバー証明書およびサブシステム証明書を自動的に承認します。 • セキュリティードメインで TPS サブシステム情報を登録および登録解除します。

必要に応じて、セキュリティードメイン管理者はセキュリティードメインおよび個別のサブシステムでアクセス制御を管理できます。たとえば、セキュリティードメイン管理者はアクセスを制限することで、KRA の管理者のみが 座有無部門の KRA を設定できるようにすることができます。

Enterprise サブシステムの管理者は、ドメインのサブシステムで操作を実行するのに十分な特権が付与されます。たとえば、エンタープライズ CA の管理者は、設定中にサブ CA 証明書を自動的に承認する特権があります。セキュリティードメイン管理者は、必要に応じてこの適切な制限を行うことができます。

15.2.2. 監査者

監査人は、署名された監査ログを表示でき、システムの動作を監査するために作成されます。監査人はサーバーを管理することはできません。

監査人は、ユーザーを **Auditors** グループに追加して、監査人の証明書をユーザーエントリーに保存することによって作成されます。監査人の証明書は、監査ログの署名に使用されるキーペアの秘密キーを暗号化するために使用されます。

サブシステムの設定時に **Auditors** グループが設定されます。設定中、このグループには監査人は割り当てられません。

監査人は、UID とパスワードを使用した単純なバインドで管理コンソールに認証されます。 認証が終わると、監査ログのみを表示できます。システムの他の部分は編集できません。

15.2.3. エージェント

エージェントは、エンドエンティティー証明書と鍵管理の特権が割り当てられているユーザーです。エージェントは、エージェントサービスインターフェイスにアクセスできます。

エージェントは、ユーザーを適切なサブシステムエージェントグループに割り当て、エージェントからの要求を処理するためにサブシステムへの SSL クライアント認証にエージェントが使用する必要のある証明書を識別することによって作成されます。各サブシステムには独自のエージェントグループがあります。

- 証明書マネージャーエージェントグループ。
- キーリカバリー認証局エージェントグループ。
- オンライン証明書ステータスマネージャーエージェントグループ。
- トークンキーサービスエージェントのグループ。
- Token Processing System Agents グループ。

各 Certificate System サブシステムには、サブシステムで定義されたロールを持つ独自のエージェントがあります。各サブシステムには少なくとも 1 つのエージェントが必要ですが、サブシステムを持つエージェントの数に制限はありません。

Certificate System は、内部データベース内でユーザーの SSL クライアント証明書をチェックして、エージェント権限を持つユーザーを特定し、認証します。

15.2.4. エンタープライズグループ

注記

実際のユーザーはこのグループに割り当てることができません。

サブシステムの設定中に、すべてのサブシステムインスタンスがセキュリティードメインに参加します。各サブシステムインスタンスには、サブシステム固有のロールがエンタープライズ管理者として自動的に割り当てられます。これらのロールはセキュリティードメインのサブシステム間で信頼できる関係を自動的に提供し、各サブシステムが他のサブシステムと効率的に対話できるようにします。たとえば、これにより、OCSP はドメイン内のすべての CA に CRL 公開情報をプッシュし、KRA は KRA コネクター情報をプッシュし、CA は CA 内で生成された証明書を自動的に承認します。

Enterprise サブシステムの管理者は、ドメインのサブシステムで操作を実行するのに十分な特権が付与されます。各サブシステムには独自のセキュリティードメインロールがあります。

- エンタープライズ CA 管理者
- エンタープライズ KRA 管理者
- エンタープライズ OCSP 管理者
- エンタープライズ TKS 管理者
- エンタープライズ TPS 管理者

また、ドメイン内のセキュリティードメイン、アクセス制御、ユーザー、および信頼関係を管理する CA インスタンス用の Security Domain Administrators のグループもあります。

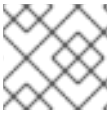
各サブシステム管理者は、セキュリティードメイン CA によって設定時に発行されたサブシステム証明書を使用した SSL クライアント認証を使用して他のサブシステムに対して認証します。

15.3. CA、OCSP、KRA、または TKS のユーザーおよびグループの管理

ユーザーが実行できる操作の多くは、ユーザーが属するグループによって決定されます。たとえば、CA のエージェントは証明書とプロファイルを管理し、管理者は CA サーバーの設定を管理します。

4 つのサブシステム (CA、OCSP、KRA、および TKS) Java 管理コンソールを使用してグループとユーザーを管理します。TPS には Web ベースの管理サービスがあり、ユーザーおよびグループは Web サービスページで設定されます。

15.3.1. グループの管理



注記

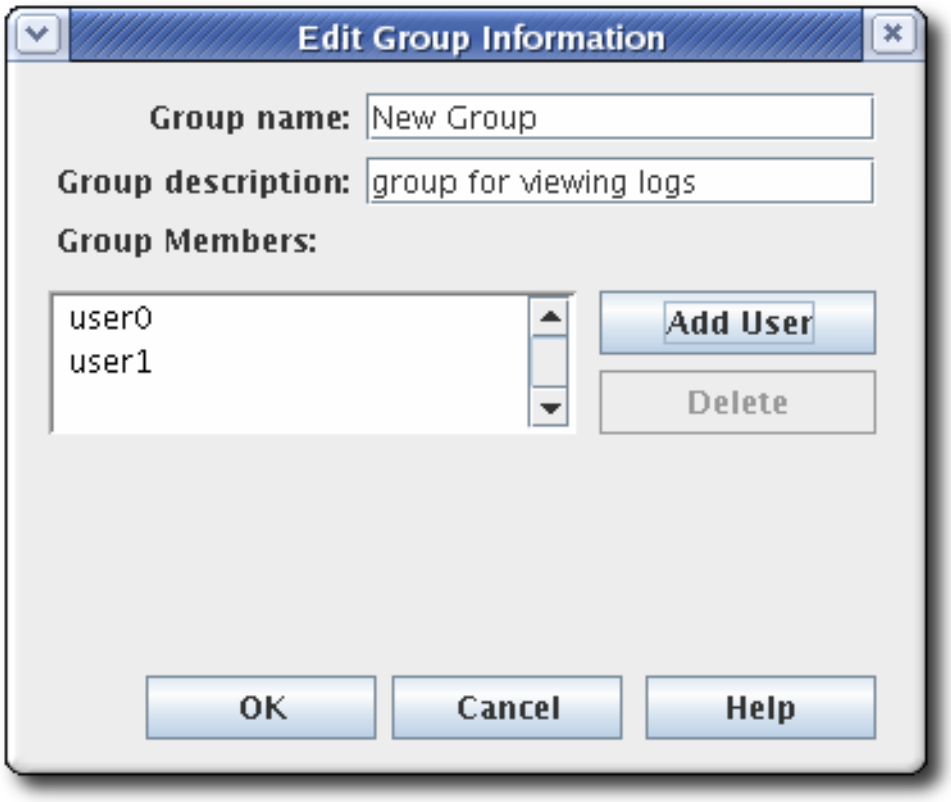
pkiconsole が非推奨になりました。

15.3.1.1. 新規グループの作成

1. 管理コンソールにログインします。

```
pkiconsole https://server.example.com:8443/subsystem_type
```

2. 左側のナビゲーションメニューから **Users and Groups** を選択します。
3. **Groups** タブを選択します。
4. **Edit** をクリックして、グループ情報を入力します。



内部データベースにすでに存在するユーザーのみを追加することが可能です。

5. ACL を編集して、グループの権限を付与します。詳細は、「ACL の編集」 を参照してください。グループの ACL に ACI が追加されていない場合、グループには Certificate System の一部にアクセスパーミッションがありません。

15.3.1.2. グループのメンバーの変更

すべてのグループからメンバーを追加または削除できます。管理者のグループには、最低でもユーザーエントリーが 1 つ必要です。

1. 管理コンソールにログインします。
2. 左側のナビゲーションツリーから **Users and Groups** を選択します。
3. **Groups** タブをクリックします。
4. 名前の一覧からグループを選択し、**Edit** をクリックします。
5. 適切な変更を加えます。
   - グループの説明を変更するには、**Group description** フィールドに新しい説明を入力します。
   - グループからユーザーを削除するには、ユーザーを選択し、**Delete** をクリックします。
   - ユーザーを追加するには、**Add User** をクリックします。ダイアログボックスから追加するユーザーを選択し、**OK** をクリックします。

15.3.2. ユーザーの管理 (管理者、エージェント、および監査者)

各サブシステムのユーザーは別々に維持されます。あるサブシステムの管理者であるからといって、その人が別のサブシステムに対する権限 (またはユーザーエントリー) を持っているとは限りません。ユーザーを設定して、ユーザー証明書を使用してサブシステムのエージェント、管理者、または監査担当者として信頼できます。

15.3.2.1. ユーザーの作成

Certificate System をインストールしたら、セットアップ時に作成したユーザーのみが存在します。本セクションでは、追加のユーザーを作成する方法を説明します。

注記

セキュリティー上の理由から、Certificate System ユーザーに個別のアカウントを作成します。

15.3.2.1.1. コマンドラインでのユーザーの作成

コマンドラインでユーザーを作成するには、以下を実行します。

1. ユーザーアカウントを追加します。たとえば、**example** ユーザーを CA に追加するには、以下を実行します。

```
# pki -d ~/.dogtag/pki-instance_name/ca/alias/ -c password -n caadmin \
    ca-user-add example --fullName "Example User"
```

```
---------------------
Added user "example"
---------------------
  User ID: example
  Full name: Example User
```

このコマンドは、**caadmin** ユーザーを使用して新規アカウントを追加します。

2. 必要に応じて、グループにユーザーを追加します。たとえば、**example** ユーザーを **Certificate Manager Agents** グループに追加するには、次のコマンドを実行します。

```
# pki -d ~/.dogtag/pki-instance_name/ -p password -n "caadmin" \
    user-add-membership example Certificate Manager Agents
```

3. 証明書要求を作成します。

- Certificate System 環境に Key Recovery Authority (KRA) が存在する場合は、以下を行います。

```
# CRMFPopClient -d ~/.dogtag/pki-instance_name/ -p password \
    -n "user_name" -q POP_SUCCESS -b kra.transport -w "AES/CBC/PKCS5Padding" \
    -v -o ~/user_name.req
```

このコマンドは、証明書署名要求 (CSR) を ***~/user_name.req*** ファイルに **CRMF** 形式を保存します。

- 証明書システム環境に Key Recovery Authority (KRA) が存在しない場合は、以下を行います。
  - NSS データベースディレクトリーを作成します。

```
# export pkiinstance=ca1
# echo ${pkiinstance}
# export agentdir=~/.dogtag/${pkiinstance}/agent1.dir
# echo ${agentdir}
# pki -d ${agentdir}/ -C ${somepwdfile} client-init
```

  - CSR を **-o** オプションで指定された PKCS-#10 フォーマットファイルに保存し、初期化された NSS データベースディレクトリーへのパスの場合は **-d**、パスワードファイルの場合は **-P** オプション、パスワードの場合は **-p**、サブジェクト DN の場合は **-n** を保存します。

```
# PKCS10Client -d ${agentdir}/ -P ${somepwdfile} -n "cn=agent1,uid=agent1" -o
${agentdir}/agent1.csr
PKCS10Client: Certificate request written into /.dogtag/ca1/agent1.dir/agent1.csr
PKCS10Client: PKCS#10 request key id written into
/.dogtag/ca1/agent1.dir/agent1.csr.keyId
```

4. 登録リクエストを作成します。

a. ***~/cmc.role_crmf.cfg*** ファイルを以下の内容で作成します。

```
#numRequests: Total number of PKCS10 requests or CRMF requests.
numRequests=1
```

```
#input: full path for the PKCS10 request or CRMF request,
#the content must be in Base-64 encoded format
#Multiple files are supported. They must be separated by space.
input=~/user_name.req

#output: full path for the CMC request in binary format
output=~/cmc.role_crmf.req

#tokenname: name of token where agent signing cert can be found (default is internal)
tokenname=internal

#nickname: nickname for agent certificate which will be used
#to sign the CMC full request.
nickname=PKI Administrator for Example.com

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
dbdir=~/.dogtag/pki-instance_name/

#password: password for cert9.db which stores the agent
#certificate
password=password

#format: request format, either pkcs10 or crmf
format=crmf
```

直前の手順で使用した環境および CSR 形式に基づいて、パラメーターを設定します。

b. 以前に作成した設定ファイルを **CMCRequest** ユーティリティーに渡して、CMC 要求を作成します。

```
# CMCRequest ~/cmc.role_crmf.cfg
```

5. CMS (CMC) 要求で Certificate Management を送信します。

a. ***~/HttpClient_role_crmf.cfg*** ファイルを以下の内容で作成します。

```
# #host: host name for the http server
host=server.example.com

#port: port number
port=8443

#secure: true for secure connection, false for nonsecure connection
secure=true

#input: full path for the enrollment request, the content must be in binary format
input=~/cmc.role_crmf.req

#output: full path for the response in binary format
output=~/cmc.role_crmf.resp

#tokenname: name of token where SSL client authentication cert can be found (default is
internal)
#This parameter will be ignored if secure=false
```

```
tokenname=internal

#dbdir: directory for cert9.db, key4.db and pkcs11.txt
#This parameter will be ignored if secure=false
dbdir=~/.dogtag/pki-instance_name/

#clientmode: true for client authentication, false for no client authentication
#This parameter will be ignored if secure=false
clientmode=true

#password: password for cert9.db
#This parameter will be ignored if secure=false and clientauth=false
password=password

#nickname: nickname for client certificate
#This parameter will be ignored if clientmode=false
nickname=PKI Administrator for Example.com

#servlet: servlet name
servlet=/ca/ee/ca/profileSubmitCMCFull
```

環境に応じてパラメーターを設定します。

b. CA に要求を送信します。

```
# HttpClient ~/HttpClient_role_crmf.cfg
Total number of bytes read = 3776
after SSLSocket created, thread token is Internal Key Storage Token
client cert is not null
handshake happened
writing to socket
Total number of bytes read = 2523
MIIJ1wYJKoZIhvcNAQcCoIIJyDCCCcQCAQMxDzANBglghkgBZQMEAgEFADAxBggr
...
The response in data format is stored in ~/cmc.role_crmf.resp
```

c. 結果を確認します。

```
# CMCResponse ~/cmc.role_crmf.resp
Certificates:
   Certificate:
      Data:
         Version:  v3
         Serial Number: 0xE
         Signature Algorithm: SHA256withRSA - 1.2.840.113549.1.1.11
         Issuer: CN=CA Signing Certificate,OU=pki-instance_name Security Domain
         Validity:
            Not Before: Friday, July 21, 2017 12:06:50 PM PDT America/Los_Angeles
            Not  After: Wednesday, January 17, 2018 12:06:50 PM PST
America/Los_Angeles
         Subject: CN=user_name
...
Number of controls is 1
Control #0: CMCStatusInfoV2
```

```
OID: {1 3 6 1 5 5 7 7 25}
BodyList: 1
Status: SUCCESS
```

6. 必要に応じて、証明書をユーザー自身の ~/**.dogtag**/**pki-instance_name**/ データベースにインポートするには、次のコマンドを実行します。

```
# certutil -d ~/.dogtag/pki-instance_name/ -A -t "u,u,u" -n "user_name certificate" -i
~/cmc.role_crmf.resp
```

7. ユーザーレコードに証明書を追加します。

   a. ユーザーのシリアル番号を検出できる証明書を一覧表示します。たとえば、証明書のサブジェクトに **example** ユーザー名が含まれる証明書を一覧表示するには、次のコマンドを実行します。

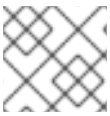
```
pki -d ~/.dogtag/pki-instance_name/ -c password -n caadmin ca-user-cert-find example
-----------------
1 entries matched
-----------------
  Cert ID: 2;6;CN=CA Signing Certificate,O=EXAMPLE;CN=PKI
Administrator,E=example@example.com,O=EXAMPLE
  Version: 2
  Serial Number: 0x6
  Issuer: CN=CA Signing Certificate,O=EXAMPLE
  Subject: CN=PKI Administrator,E=example@example.com,O=EXAMPLE
----------------------------
Number of entries returned 1
```

   次の手順では、証明書のシリアル番号が必要です。

   b. シリアル番号を使用して、証明書リポジトリーから Certificate System データベースのユーザーアカウントに証明書を追加します。たとえば、CA ユーザーの場合は以下を実行します。

```
pki -c password -n caadmin ca-user-cert-add example --serial 0x6
```

15.3.2.1.2. コンソールを使用したユーザーの作成



注記

pkiconsole が非推奨になりました。

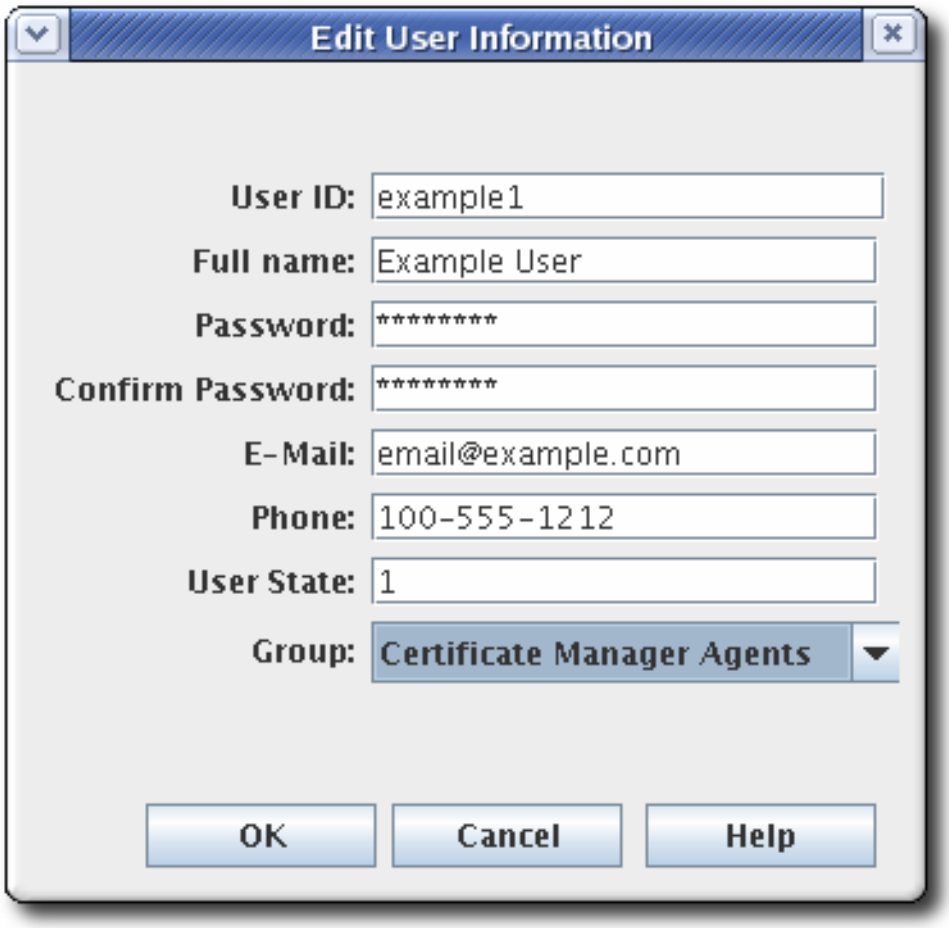
PKI コンソールを使用してユーザーを作成するには、次のコマンドを実行します。

1. 管理コンソールにログインします。

```
pkiconsole https://server.example.com:8443/subsystem_type
```

2. **Configuration** タブで、**Users and Groups** を選択します。**Add** をクリックします。

3. **Edit User Information** ダイアログに情報を入力します。



情報のほとんどは、ユーザー名、メールアドレス、パスワードなどの標準のユーザー情報です。このウィンドウには、**User State** と呼ばれるフィールドも含まれ、このフィールドには、ユーザーに関する追加情報を追加するのに使用される文字列を含めることができます。ほとんどの場合、このフィールドは、アクティブユーザーであるかどうかを確認できます。

4. ユーザーが属するグループを選択します。ユーザーのグループメンバーシップは、ユーザーが持つ特権を決定します。エージェント、管理者、および監査人を適切なサブシステムグループに割り当てます。

5. ユーザーの証明書を保存します。

    a. CA エンドエンティティーサービスページでユーザー証明書を要求します。

    b. ユーザープロファイルに対して自動登録が設定されていない場合は、証明書要求を承認します。

    c. 通知メールで提供される URL を使用して証明書を取得し、base-64 でエンコードされた証明書をローカルファイルまたはクリップボードにコピーします。

    d. 新しいユーザーエントリーを選択し、**Certificates** をクリックします。

    e. **Import** をクリックし、Base-64 でエンコードされた証明書に貼り付けます。

15.3.2.2. 証明書システムユーザー証明書の変更

1. 管理コンソールにログインします。

2. **Users and Groups** を選択します。

3. ユーザー ID の一覧から編集するユーザーを選択し、**Certificates** をクリックします。

4. **Import** をクリックして、新しい証明書を追加します。

5. **Import Certificate** ウィンドウで、テキストエリアに新しい証明書を貼り付けます。**-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** マーカー行を含めます。

15.3.2.3. 管理者、エージェント、および監査ユーザー証明書の更新

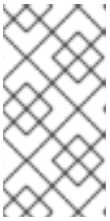
証明書を更新する方法は 2 つあります。証明書を **再生成** すると、元の鍵と元のプロファイルと要求を取得し、新しい有効期間と有効期限で同一の鍵を再作成します。証明書の**キーを再入力** すると、最初の証明書要求が元のプロファイルに再送信されますが、新しいキーペアが生成されます。管理者証明書は、キーを再入力することで更新できます。

各サブシステムには、サブシステムの作成時に作成されたブートストラップユーザーがあります。デフォルトの更新プロファイルの 1 つを使用して、元の証明書の有効期限が切れる前に、このユーザーに新しい証明書を要求できます。

管理ユーザーの証明書は、元の証明書のシリアル番号を使用して、エンドユーザー登録フォームで直接更新できます。

1. 「証明書ベースの更新」 の説明に従って、CA のエンドユーザーフォームで管理ユーザー証明書を更新します。これは、最初に発行した証明書 (またはそのクローン) と同じ CA である必要があります。

   エージェント証明書は、エンドエンティティーページで証明書ベースの更新フォームを使用して更新できます。**Self-renew user SSL client certificate**。このフォームは、ブラウザーの証明書ストアに保存されている証明書を直接認識して更新します。

   注記

   「certutil を使用した証明書の更新」 で説明されているように、**certutil** を使用して証明書を更新することもできます。ブラウザーに保存されている証明書を使用して更新を開始するのではなく、**certutil** は元のキーで入力ファイルを使用します。

2. 更新されたユーザー証明書を内部 LDAP データベースのユーザーエントリーに追加します。

   a. サブシステムのコンソールを開きます。

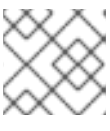
   ```
   pkiconsole https://server.example.com:admin_port/subsystem_type
   ```

   b. 設定 | ユーザーとグループ | ユーザー | 管理 | 証明書 | インポート

   c. **Configuration** タブで、**Users and Groups** を選択します。

   d. **Users** タブで、更新された証明書でユーザーエントリーをダブルクリックして、**Certificates** をクリックします。

   e. **Import** をクリックし、Base-64 でエンコードされた証明書に貼り付けます。

   注記

   pkiconsole が非推奨になりました。

これは、**ldapmodify** を使用して、**uid=admin,ou=people,dc=**subsystem-base-DN など、ユーザーエントリーの ***userCertificate*** 属性を置き換え、内部の LDAP データベースでユーザーエントリーに直接更新した証明書を追加しました。

15.3.2.4. 期限切れの管理者、エージェント、および監査者のユーザー証明書の更新

有効なユーザー証明書の有効期限がすでに切れている場合、Web サービスページも、認証が必要な **pki** コマンドラインツールも使用できなくなります。このようなシナリオでは、**pki-server cert-fix** コマンドを使用して、期限切れの証明書を更新できます。

続行する前に、次のことを確認してください。

- 有効な CA 証明書があります。
- root 権限がある

手順15.1 期限切れの管理者、エージェント、および監査者のユーザー証明書の更新

1. セルフテストを無効にします。
   - 次のコマンドを実行します。

     ```
     # pki-server selftest-disable -i PKI_instance
     ```

   - または、CA の **CS.cfg** ファイルから次の行を削除し、CA サブシステムを再起動します。

     ```
     selftests.container.order.startup=CAPresence:critical, SystemCertsVerification:critical
     ```

2. クライアントの NSS データベースで期限切れの証明書を確認し、証明書のシリアル番号 (証明書 ID) を見つけます。
   a. ユーザー証明書をリスト表示します。

      ```
      # certutil -L -d /root/nssdb/
      ```

   b. 更新する期限切れの証明書のシリアル番号を取得します。

      ```
      # certutil -L -d /root/nssdb/ -n Expired_cert | grep Serial
          Serial Number: 16 (0x10)
      ```

3. 証明書を更新します。ローカル LDAP サーバーには、LDAP Directory Manager のパスワードが必要です。

   ```
   # pki-server cert-fix --ldap-url ldap://host389 --agent-uid caadmin -i PKI_instance -p
   PKI_https_port --extra-cert 16
   ```

4. セルフテストを再度有効にします。
   - 次のコマンドを実行します。

     ```
     # pki-server selftest-enable -i PKI_instance
     ```

   - または、次の行を CA の **CS.cfg** ファイルに追加して、CA サブシステムを再起動します。

```
selftests.container.order.startup=CAPresence:critical, SystemCertsVerification:critical
```

証明書の更新に成功したことを確認するには、次のコマンドを実行して、証明書に関する十分な情報を表示できます。

```
# pki ca-cert-find
```

属性、拡張機能、公開鍵モジュラス、ハッシュなどを含む特定の証明書の完全な詳細を表示するには、次を実行することもできます。

```
# pki ca-cert-show 16 --pretty
```

15.3.2.5. 証明書システムユーザーの削除

ユーザーは内部データベースから削除できます。内部データベースからユーザーを削除すると、そのユーザーが属するすべてのグループから削除されます。特定のグループからユーザーを削除するには、グループメンバーシップを変更します。

以下の手順を実行して、内部データベースから特権ユーザーを削除します。

1. 管理コンソールにログインします。
2. 左側のナビゲーションメニューから **Users and Groups** を選択します。
3. ユーザー ID の一覧からユーザーを選択して、**Delete** をクリックします。
4. プロンプトが表示されたら、削除を確認します。

15.4. TPS のユーザーの作成および管理

TPS ユーザーに対して定義された **ロール** が 3 つあります。これは、TPS のグループとして機能します。

- **エージェント** (トークンのステータスの設定やトークンポリシーの変更など、実際のトークン管理操作を実行するエージェント)
- **管理者** (TPS サブシステムのユーザーを管理し、トークンの制御を制限)
- **オペレーター** (管理制御がないが、TPS からトークン、証明書、およびアクティビティーを表示および一覧表示できる)

TPS には、追加のグループを追加できません。

すべての TPS サブシステムユーザーは、証明書を含む LDAP ディレクトリーデータベースに対して認証され (TPS の Web サービスにアクセスするには証明書ベースの認証が必要なため)、認証プロセスは TPS グループエントリー **ou=TUS Agents**、**ou=TUS Administrators**、**ou=TUS Agents** を使用して、Apache の **mod_tokendb** モジュールを使用して、ユーザーが所属するロールを表示していることを確認します。

TPS のユーザーは、Web UI または CLI で追加および管理されます。Web UI は **https://*server.example.com*:8443/tps/ui/** でアクセスできます。

Web UI または CLI を使用するには、TPS 管理者はユーザー証明書を使用して認証する必要があります。

15.4.1. ユーザーの一覧表示および検索

15.4.1.1. Web UI での操作

Web UI でユーザーを一覧表示するには、以下を実行します。

1. **Accounts** タブをクリックします。
2. **Users** メニュー項目をクリックします。ユーザーの一覧が表示されます。
3. 特定のユーザーを検索するには、検索フィールドにキーワードを入力して **Enter** を押します。すべてのユーザーを再度一覧表示するには、キーワードを削除して **Enter** キーを押します。

15.4.1.2. コマンドラインでの操作

CLI のユーザーを一覧表示するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-find
```

CLI からユーザー情報を表示するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-show username
```

15.4.2. ユーザーの追加

15.4.2.1. Web UI での操作

Web UI でユーザーを追加するには、以下を実行します。

1. **Accounts** タブをクリックします。
2. **Users** メニュー項目をクリックします。
3. **Users** ページの **Add** ボタンをクリックします。
4. ユーザー ID、フルネーム、および TPS プロファイルを入力します。
5. **Save** ボタンをクリックします。

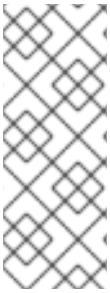
15.4.2.1.1. コマンドラインでの操作

CLI からユーザーを追加するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-add username --
fullName full_name
```

15.4.3. ユーザーのプロファイルの設定

TPS プロファイルは CA プロファイルとほぼ似ており、異なるタイプのトークンを処理するルールを定義します。プロファイルは、CUID などのトークンの特徴に基づいて自動的にトークンに割り当てられます。ユーザーは、割り当てられたプロファイルのトークンのみを表示できます。



注記

ユーザーは、トークン操作とトークンの両方など、設定されたプロファイルに関連するエントリーのみを表示できます。管理者が TPS で設定されたすべてのトークンを検索および管理できるようにするには、管理者ユーザーエントリーを **All profiles** に設定する必要があります。ユーザーに対して特定のプロファイルを設定することは、オペレーターおよびエージェントへのアクセスを特定のユーザーまたはトークンタイプに制御する簡単な方法です。

トークンプロファイルは、トークンに適用されるポリシーおよび設定のセットです。トークンプロファイルは、CCUID 範囲など、トークン自体の属性に基づいて自動的にトークンにマップされます。トークンプロファイルは、CA プロファイルディレクトリーに他の証明書プロファイルとして作成され、TPS 設定ファイル **CS.cfg** に追加されて、CA のトークンプロファイルをトークンタイプにマッピングします。トークンマッピングの設定は、「Mapping Resolver の設定」 で説明されています。

Web UI でユーザープロファイルを管理するには、以下を実行します。

1. **Accounts** タブをクリックします。
2. **Users** メニュー項目をクリックします。
3. 変更するユーザーのユーザー名をクリックします。
4. **Edit** のリンクをクリックします。
5. **TPS Profile** フィールドにプロファイル名をコンマで区切って入力するか、**All Profiles** を入力します。
6. **Save** ボタンをクリックします。

15.4.4. ユーザーロールの管理

ロールは、TPS 内の単なるグループです。各ロールは、TPS サービスページのさまざまなタブを表示できます。このグループは編集されているため、ユーザーのロール割り当てを追加または削除できます。

ユーザーは、複数のロールまたはグループに属することができます。たとえば、ブートストラップユーザーは 3 つのグループすべてに属します。

15.4.4.1. Web UI での操作

Web UI からグループメンバーを管理するには、以下を実行します。

1. **Accounts** タブをクリックします。
2. **Groups** メニュー項目をクリックします。
3. TPS エージェントなど、変更するグループの名前をクリックします。
4. このグループにユーザーを追加するには、以下を実行します。
    a. **Add** ボタンをクリックします。
    b. ユーザー ID を入力します。
    c. **Add** ボタンをクリックします。

5. このグループからユーザーを削除するには、次を実行します。

    a. ユーザーの横にあるチェックボックスを選択します。

    b. **Remove** ボタンをクリックします。

    c. **OK** ボタンをクリックします。

15.4.4.2. コマンドラインでの操作

CLI からグループの一覧を表示するには、次のコマンドを実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-find
```

CLI からグループメンバーを一覧表示するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-find
group_name
```

CLI からユーザーをグループに追加するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-add
group_name user_name
```

CLI からグループからユーザーを削除するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-group-member-del
group_name user_name
```

15.4.5. ユーザー証明書の管理

ユーザー証明書は CLI から管理できます。

- ユーザー証明書を一覧表示するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-find
user_name
```

- 証明書をユーザーに追加するには、以下を実行します。

1. 新規ユーザーのユーザー証明書を取得します。証明書の要求および送信については、5章*証明書の要求、登録、および管理* で説明されています。

重要

TPS 管理者が署名証明書が必要です。使用する推奨プロファイルは、手動のユーザー署名および暗号化証明書の登録です。

2. 次のコマンドを実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-add
user_name --serial cert_serial_number
```

- ユーザーから証明書を削除するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-cert-del
user_name cert_id
```

15.4.6. TPS エージェントおよび管理者証明書の更新

証明書を **再生成** すると、元の鍵と元のプロファイルと要求を取得し、新しい有効期間と有効期限で同一の鍵を再作成します。

TPS には、サブシステムの作成時に作成されたブートストラップユーザーがあります。デフォルトの更新プロファイルの 1 つを使用して、元の証明書の有効期限が切れる前に、このユーザーに新しい証明書を要求できます。

管理ユーザーの証明書は、元の証明書のシリアル番号を使用して、エンドユーザー登録フォームで直接更新できます。

1. 「証明書ベースの更新」 の説明に従って、CA のエンドユーザーフォームでユーザー証明書を更新します。これは、最初に発行した証明書 (またはそのクローン) と同じ CA である必要があります。

   エージェント証明書は、エンドエンティティーページの証明書ベースの更新フォーム (**Self-renew user SSL client certificate**) を使用して更新できます。このフォームは、ブラウザーの証明書ストアに保存されている証明書を直接認識して更新します。

   注記

   「certutil を使用した証明書の更新」 で説明されているように、**certutil** を使用して証明書を更新することもできます。ブラウザーに保存されている証明書を使用して更新を開始するのではなく、**certutil** は元のキーで入力ファイルを使用します。

2. 新しい証明書をユーザーに追加し、「ユーザー証明書の管理」 の説明に従って古い証明書を削除します。

15.4.7. ユーザーの削除

警告

最後 のユーザーアカウントを削除することができ、この操作は取り消せません。削除するユーザーについて、十分に注意してください。

Web UI でユーザーを削除するには、以下を実行します。

1. **Accounts** タブをクリックします。
2. **Users** メニュー項目をクリックします。
3. 削除するユーザーの横にあるチェックボックスを選択します。

4. **Remove** ボタンをクリックします。

5. **OK** ボタンをクリックします。

CLI からユーザーを削除するには、以下を実行します。

```
pki -d client_db_dir -c client_db_password -n admin_cert_nickname tps-user-del user_name
```

15.5. ユーザーのアクセス制御の設定

承認 は、ユーザーが操作を実行できるかどうかを確認するメカニズムです。許可ポイントは、許可チェックを必要とする特定の操作グループで定義されます。

15.5.1. アクセス制御について

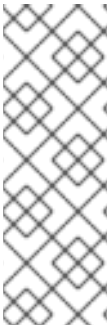
アクセス制御リスト (ACL) は、サーバー操作への承認を指定するメカニズムです。承認チェックが実行される操作ごとに ACL が存在します。ACL に追加の操作を追加できます。

ACL には、読み取りや変更などの操作を具体的に許可または拒否する **アクセス制御命令** (ACI) が含まれます。ACI にはエバリュエーターの式も含まれます。ACL のデフォルトの実装は、ユーザー、グループ、および IP アドレスのみを、可能なエバリュエータータイプとして指定します。ACL の各 ACI は、アクセスが許可または拒否されるかどうか、特定の Operator が許可または拒否されているか、および操作を実行するためのユーザー、グループ、または IP アドレスが許可または拒否されるかどうかを指定します。

Certificate System ユーザーの特権は、ユーザーがメンバーであるグループ、ユーザー自身、またはユーザーの IP アドレスに関連付けられているアクセス制御リスト (ACL) を変更することによって変更されます。新規グループは、そのグループをアクセス制御リストに追加することで、アクセス制御リストに割り当てられます。たとえば、ログ **LogAdmins** の表示が許可される管理者用の新規グループは、このグループの読み取りまたは修正を許可するためにログに関連する ACL に追加できます。このグループが他の ACL に追加されない場合、このグループのメンバーはログにのみアクセスできます。

ACL の ACI エントリーを編集して、ユーザー、グループ、または IP アドレスへのアクセスが変更されます。ACL インターフェイスでは、各 ACI が独自の行に表示されます。このインターフェイスウィンドウで、ACI の構文は以下のとおりです。

```
allow|deny (operation) user|group|IP="name"
```



注記

IP アドレスは、IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:**および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。

たとえば、以下は ACI で、管理者は読み取り操作を実行できます。

```
allow (read) group="Administrators"
```

ACI には、複数の操作またはアクションを設定できます。操作は、両側にスペースを入れずにコンマで区切ります。以下に例を示します。

```
allow (read,modify) group="Administrators"
```

ACI は、2 つのパイプ記号で区切ることにより、複数のグループ、ユーザー、または IP アドレスを、両側にスペースがある状態で指定することができます (||)。以下に例を示します。

```
allow (read) group="Administrators" || group="Auditors"
```

管理コンソールは ACI を作成または変更できます。このインターフェイスは、**Allow and Deny** フィールドで操作を許可するかどうか、**Operations** フィールドで可能な操作を設定し、次に **Syntax** フィールドでグループ、ユーザー、または IP アドレスを一覧表示します。

ACI は指定されたグループ、ユーザー ID、または IP アドレスの操作を許可または拒否できます。通常、アクセスを拒否するために ACI を作成する必要はありません。ユーザー ID、グループ、または IP アドレスを含む allow ACI がない場合、グループ、ユーザー ID、または IP アドレスへのアクセスは拒否されます。

注記

ユーザーがリソースのどの操作にも明示的に許可されていない場合、このユーザーは拒否されます。アクセスを拒否する必要はありません。

たとえば、ユーザー JohnB は **Administrators** グループのメンバーです。ACL には以下の ACL のみがある場合は、allow ACI に一致しないため、JohnB はすべてのアクセスを拒否します。

```
Allow (read,modify) group="Auditors" || user="BrianC"
```

通常、deny ステートメントを含める必要はありません。ただし、指定すると便利な場合もあります。たとえば、**Administrators** グループのメンバーである **JohnB** が唯一実行されています。ユーザーをすぐに削除できない場合は、特に **JohnB** へのアクセスを拒否する必要がある場合があります。もう 1 つの状況は、ユーザー **BrianC** が管理者であるが、一部のリソースを変更する権限を持たない場合です。**Administrators** グループはこのリソースにアクセスする必要があるため、**BrianC** はこのユーザーアクセスを拒否する ACI を作成して、アクセスを拒否することができます。

許可される権限は、ACI が操作の実行を許可または拒否することで ACI が制御する操作です。ACL に設定できるアクションは ACL とサブシステムによって異なります。定義できる 2 つの一般的な操作は、読み取りと変更です。

ACI エディターの構文フィールドは、式にエバリュエーターを設定します。エバリュエーターは、グループ、名前、および IP アドレス (IPv4 アドレスと IPv6 アドレスの両方) を指定できます。これらは、同一 (**=**) または非同一 (**!=**) として設定されたエンティティーの名前とともに指定されます。

ACL にグループを追加する構文は **group="groupname"** です。グループを除外する構文は **group!="groupname"** で、named グループ以外のグループを許可します。以下に例を示します。

```
group="Administrators" || group!="Auditors"
```

アスタリスク (*) などのワイルドカード文字を使用するなど、正規表現を使用してグループを指定することもできます。以下に例を示します。

```
group="* Managers"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

ACL にユーザーを追加する構文は **user="userID"** です。ユーザーを除外する構文は **user!="userID"** です。これにより、名前が指定されたユーザー ID 以外のユーザー ID も使用できます。以下に例を示します。

```
user="BobC" || user!="JaneK"
```

すべてのユーザーを指定するには、**anybody** の値を指定します。以下に例を示します。

```
user="anybody"
```

正規表現を使用して、アスタリスク (*) などのワイルドカード文字を使用するなど、ユーザー名を指定することもできます。以下に例を示します。

```
user="*johnson"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

ACL に IP アドレスを追加する構文は **ipaddress="ipaddress"** です。ACL から ID アドレスを除外する構文は **ipaddress!="ipaddress"** です。IP アドレスは数値を使用して指定します。DNS 値は許可されません。以下に例を示します。

```
ipaddress="12.33.45.99"
ipaddress!="23.99.09.88"
```

IP アドレスは、上記のように IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスには、ネットマスクが **n.n.n.n** または **n.n.n.n,m.m.m.m** の形式があります。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。以下に例を示します。

```
ipaddress="0:0:0:0:0:0:13.1.68.3"
```

正規表現を使用して、アスタリスク (*) などのワイルドカード文字を使用するなど、IP アドレスを指定することもできます。以下に例を示します。

```
ipaddress="12.33.45.*"
```

サポートされる正規表現パターンの詳細は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.htmlを参照してください。

各値を 2 つのパイプ文字 (||) で区切り、両側にスペースを入れることで、複数の値を持つ文字列を作成できます。以下に例を示します。

```
user="BobC" || group="Auditors" || group="Administrators"
```

15.5.2. サブシステムのアクセス制御設定の変更

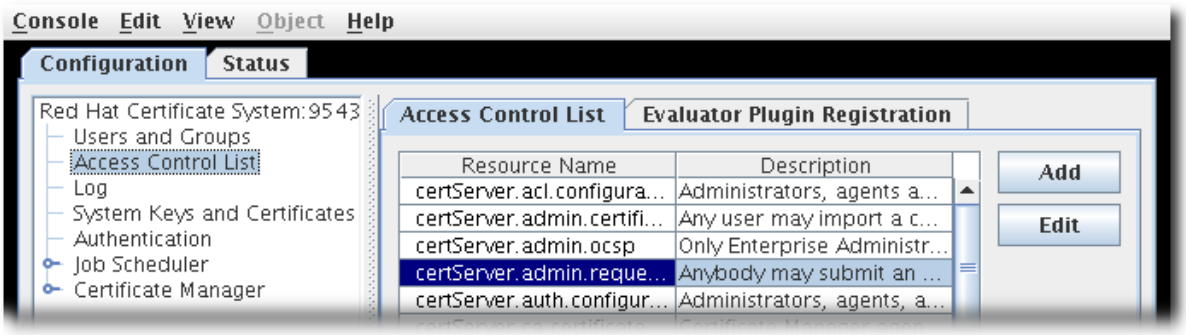
CS.cfg ファイルを編集してこの機能を設定する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『サブシステムのアクセス制御設定の変更』を参照してください。

15.5.3. ACL の追加

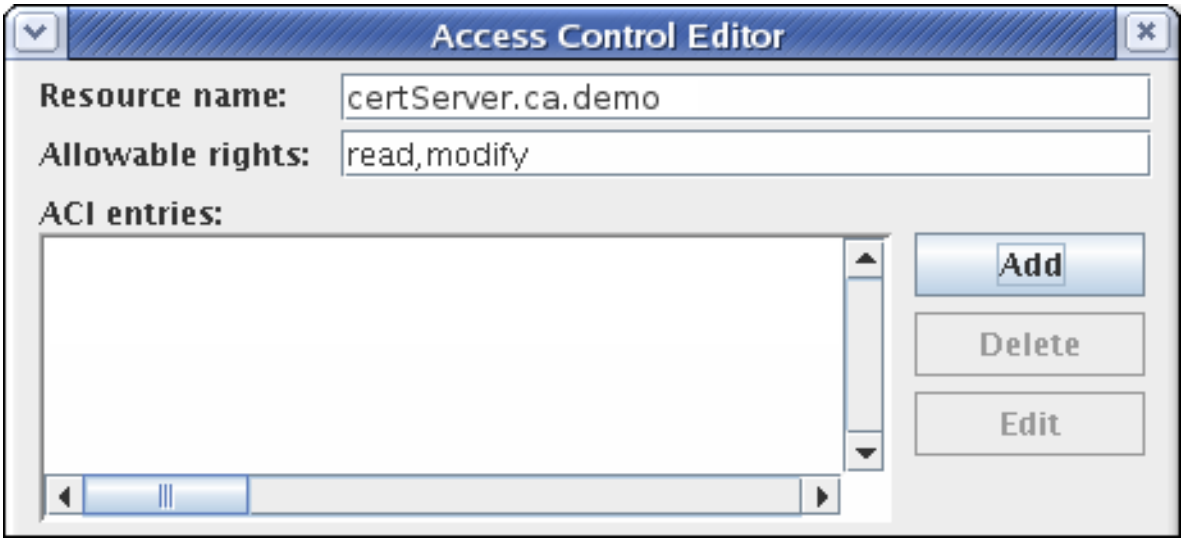
ACL は内部データベースに保存され、管理コンソールでのみ変更できます。

新しい ACL を追加するには、以下を実行します。

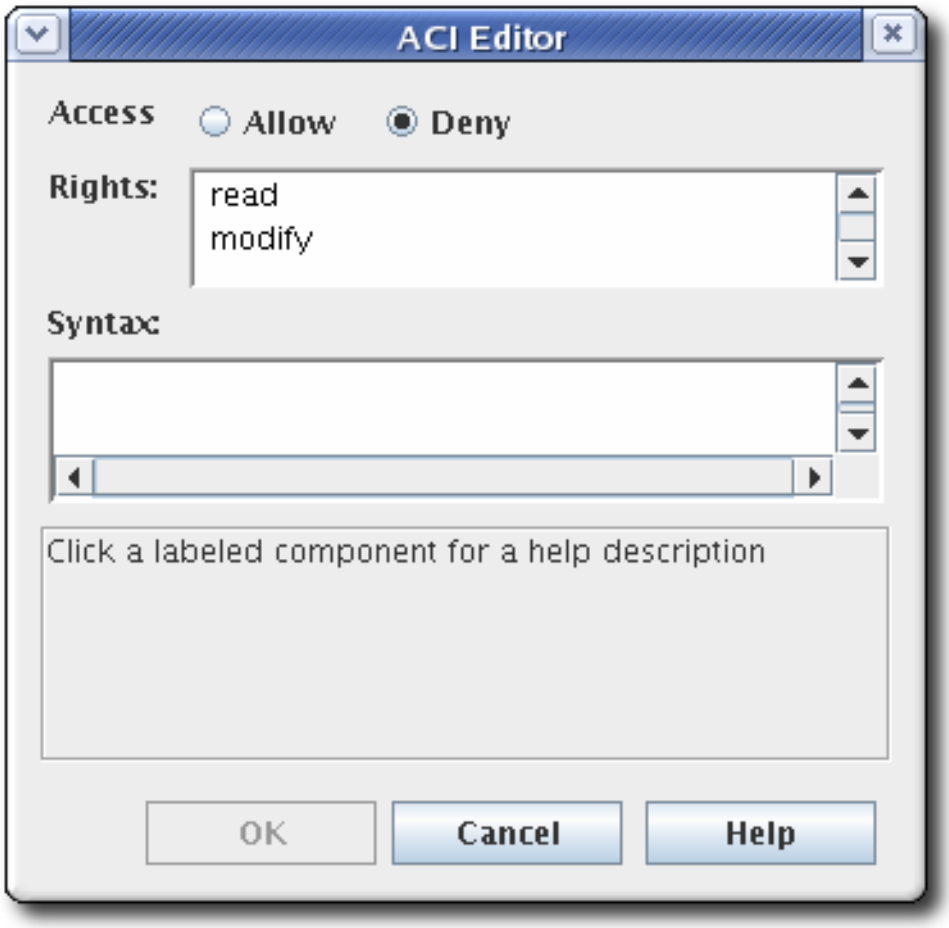
1. 管理コンソールにログインします。
2. **Access Control List** を選択します。



3. **Add** をクリックして、**Access Control Editor** を開きます。
4. **Resource name** および **Available rights** フィールドに入力します。



5. アクセス制御指示 (ACI) を追加するには、**Add** をクリックし、ACI 情報を提供します。



a. 指定したグループ、ユーザー、または IP アドレスへの操作を許可または拒否するには、**Access** フィールドから allow または deny ラジオボタンを選択します。アクセスの許可または拒否に関する詳細は、「アクセス制御について」を参照してください。

b. 権限を設定します。利用できるオプションは、**read** および **modify** です。両方を選択するには、エントリーの選択中に **Ctrl** ボタンまたは **Shift** ボタンを保持します。

c. **Syntax** フィールドでアクセスを許可または拒否されるユーザー、グループ、または IP アドレスを指定します。構文の詳細は、「アクセス制御について」を参照してください。

6. **OK** をクリックして、**Access Control Editor** 画面に戻ります。

7. **OK** をクリックして ACI を保存します。

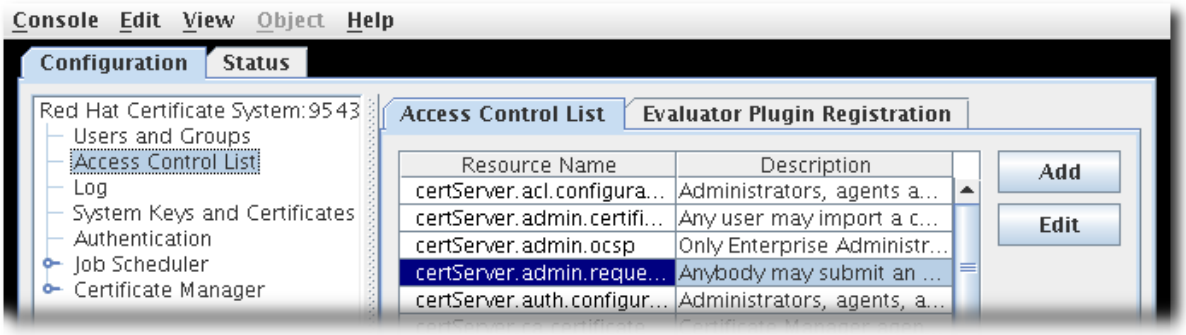
15.5.4. ACL の編集

ACL は内部データベースに保存され、管理コンソールでのみ変更できます。

既存の ACL を編集するには、以下を実行します。

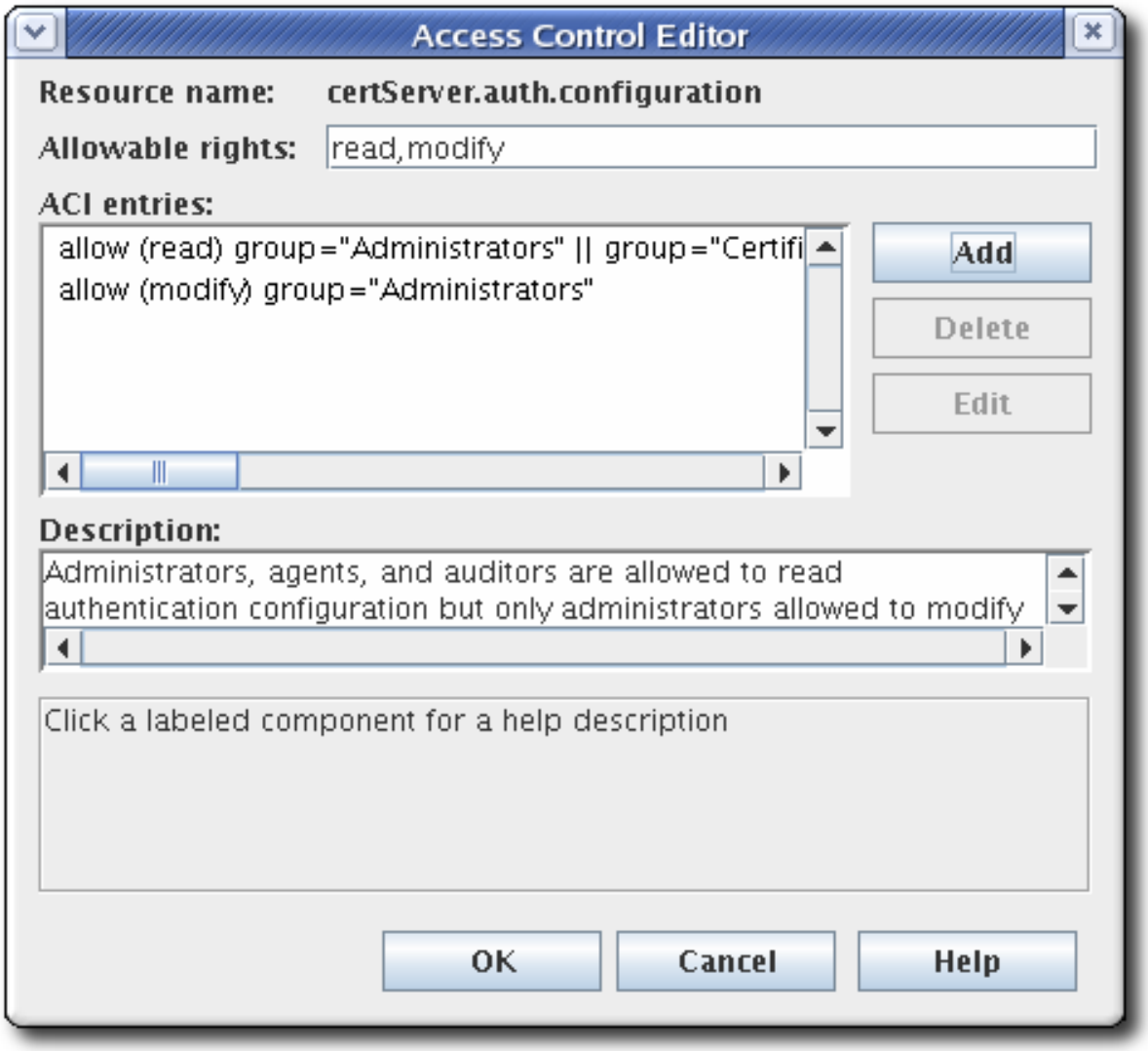
1. 管理コンソールにログインします。

2. 左側のナビゲーションメニューで、**Access Control List** を選択します。



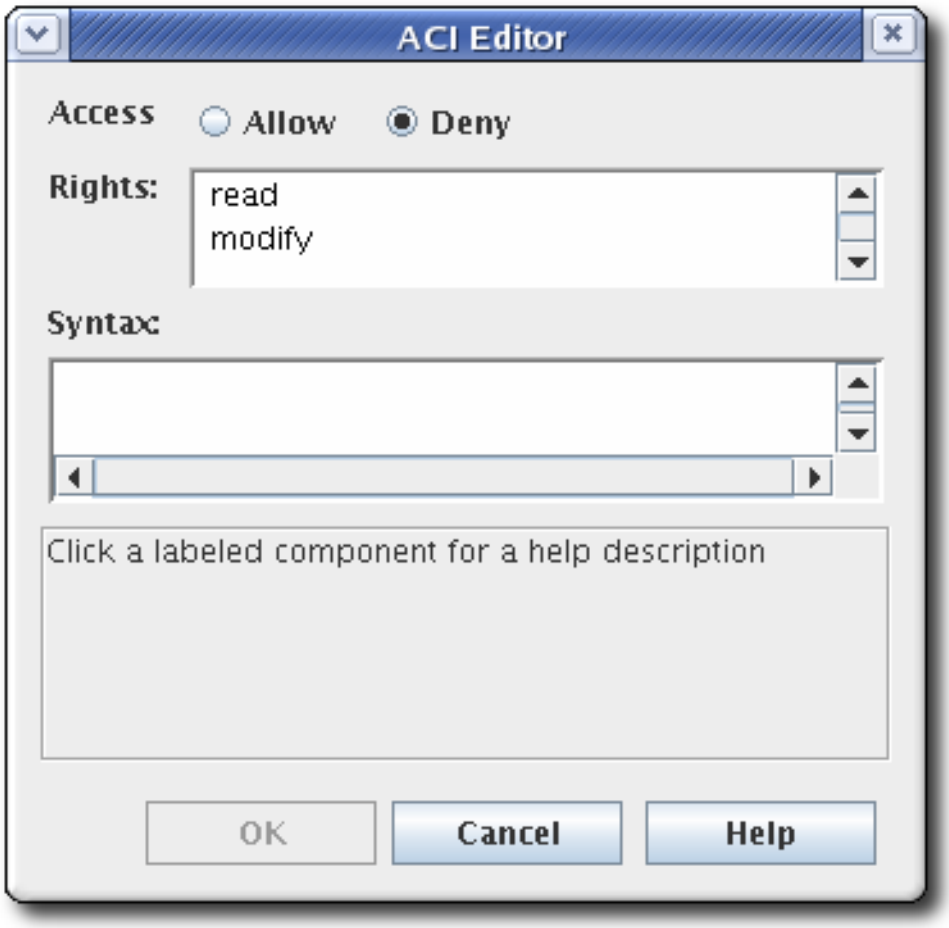
3. リストから編集する ACL を選択し、**Edit** をクリックします。

   アクセス制御エディター ウィンドウで ACL が開きます。



4. ACI を追加するには、**Add** をクリックし、ACI 情報を指定します。

   ACI を編集するには、**ACL Editor** 画面の **ACI entries** テキスト領域で ACI を選択します。**Edit** をクリックします。



a. 指定したグループ、ユーザー、または IP アドレスへの操作を許可または拒否するには、**Access** フィールドから allow または deny ラジオボタンを選択します。アクセスの許可または拒否に関する詳細は、「アクセス制御について」を参照してください。

b. アクセス制御の権限を設定します。オプションは **read** および **modify** です。両方を設定するには、**Ctrl** ボタンまたは **Shift** ボタンを使用します。

c. **Syntax** フィールドでアクセスを許可または拒否されるユーザー、グループ、または IP アドレスを指定します。構文の詳細は、「アクセス制御について」を参照してください。

第16章 サブシステムログの設定

Certificate System サブシステムは、管理、サーバーがサポートするプロトコルを使用した通信、およびサブシステムで使用されるその他のさまざまなプロセスなどのアクティビティーに関連するイベントを記録するログファイルを作成します。サブシステムインスタンスが実行中に、それが管理するすべてのコンポーネントの情報およびエラーメッセージのログを保持します。また、Apache および Tomcat の Web サーバーはエラーを生成し、ログにアクセスします。

各サブシステムインスタンスは、インストール、監査、およびその他のログに記録された機能に対する独自のログファイルを維持します。

ログプラグインモジュールは、Java™ クラスとして実装され、設定フレームワークに登録されるリスナーです。

監査ログを除くすべてのログファイルとローテーションされたログファイルは、**pkispawn** によるインスタンスの作成時に、**pki_subsystem_log_path** に指定されているすべてのディレクトリーに配置されます。 通常の監査ログは他のログタイプとともにログディレクトリーに置かれ、署名された監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/signedAudit** に書き込まれます。ログのデフォルトの場所を変更するには、設定を変更してください。

16.1. CERTIFICATE SYSTEM ログについて

Certificate System サブシステムは、サブシステムのタイプ、サービスのタイプ、および個々のログ設定に応じて特定の情報を提供する、いくつかの異なる種類のログを保持します。インスタンスに保持できるログの種類は、そのサブシステムの種類により異なります。

16.1.1. 署名付き監査ログ

Certificate System は、証明書の要求、発行、取り消し、CRL の公開など、すべてのイベントの監査ログを維持します。これらのログは署名されます。これにより、承認されたアクセスやアクティビティーの検出が可能になります。その後、外部監査人は必要に応じてシステムを監査できます。割り当てられた監査ユーザーアカウントは、署名された監査ログを表示できる唯一のアカウントです。このユーザーの証明書は、ログを署名および暗号化するために使用されます。監査ロギングは、ログに記録されるイベントを指定するよう設定されます。

署名付き監査ログは **/var/log/pki/instance_name/subsystem_name/signedAudit** に書き込まれます。ただし、設定を変更することで、ログのデフォルトの場所を変更できます。

詳細は、「署名監査ログの使用」 を参照してください。

16.1.2. デバッグログ

デフォルトで有効になっているデバッグログは、さまざまな程度と種類の情報とともに、すべてのサブシステムに対して維持されます。

デバッグログには、実行されるプラグインとサーブレット、接続情報、サーバーの要求と応答のメッセージなど、サブシステムによって実行されるすべての操作に関する非常に具体的な情報が含まれています。

デバッグログに記録されるサービスの一般的なタイプについては、「ログに記録されるサービス」 に簡単に説明します。これらのサービスには、承認要求、証明書要求の処理、証明書ステータスの確認、鍵のアーカイブおよび復元、Web サービスへのアクセスが含まれます。

サブシステムのプロセスに関する詳細情報の CA、OCSP、KRA、および TKS のデバッグログ。各ログエントリーの形式は以下のとおりです。

> [*date:time*] [*processor*]: *servlet*: *message*

メッセージ はサブシステムから返されたメッセージになるか、サブシステムに送信された値を含めることができます。

たとえば、TKS は、LDAP サーバーに接続するためにこのメッセージを記録します。

> [10/Jun/2020:05:14:51][main]: Established LDAP connection using basic authentication to host localhost port 389 as cn=Directory Manager

processor は **main** で、**message** は LDAP 接続に関するサーバーからメッセージであり、サーブレットはありません。

一方、CA は、証明書操作およびサブシステム接続に関する情報を記録します。

> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.requestowner$ value=KRA-server.example.com-8443

この場合、**プロセッサー** は CA のエージェントポート上の HTTP プロトコルですが、プロファイルを処理するサーブレットを指定し、profile パラメーター (リクエストのサブシステム所有者) とその値 (KRA が要求を開始した) を示す **メッセージ** が含まれます。

> **例16.1 CA 証明書要求ログメッセージ**
>
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.profileapprovedby$ value=admin
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.cert_request$ value=MIIBozCCAZ8wggEFAgQqTfoHMIHHgAECpQ4wDDEKMAgGA1UEAxMBeKaBnzANBgkqhkiG9w0BAQEFAAOB...
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.profile$ value=true
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.cert_request_type$ value=crmf
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.requestversion$ value=1.0.0
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.req_locale$ value=en
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.requestowner$ value=KRA-server.example.com-8443
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.dbstatus$ value=NOT_UPDATED
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.subject$ value=uid=jsmith, e=jsmith@example.com
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.requeststatus$ value=begin
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.auth_token.user$ value=uid=KRA-server.example.com-8443,ou=People,dc=example,dc=com
> [06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.req_key$ value=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxvkLP^M
> HcN0cusY7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChV^M
> k9HYDhmJ8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5I+2gE9PUznxJaM^M
> HTmlOqm4HwFxzy0RRQIDAQAB

```
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.authmgrinstname$ value=raCertAuth
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.uid$ value=KRA-server.example.com-8443
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.userid$ value=KRA-server.example.com-8443
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.requestor_name$ value=
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.profileid$
value=caUserCert
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.userdn$ value=uid=KRA-server.example.com-
4747,ou=People,dc=example,dc=com
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet: key=$request.requestid$
value=20
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.auth_token.authtime$ value=1212782378071
[06/Jun/2020:14:59:38][http-8443;-Processor24]: ProfileSubmitServlet:
key=$request.req_x509info$
value=MIICIKADAgECAgEAMA0GCSqGSIb3DQEBBQUAMEAxHjAcBgNVBAoTFVJlZGJ1ZGNv^M

bXB1dGVyIERvbWFpbjEeMBwGA1UEAxMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4X^M
DTA4MDYwNjE5NTkzOFoXDTA4MTIwMzE5NTkzOFowOzEhMB8GCSqGSIb3DQEJARYS^M
anNtaXRoQGV4YW1wbGUuY29tMRYwFAYKCZImiZPyLGQBARMGanNtaXRoMIGfMA0G^M
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDreuEsBWq9WuZ2MaBwtNYxvkLPHcN0cusY^M
7gxLzB+XwQ/VsWEoObGldg6WwJPOcBdvLiKKfC605wFdynbEgKs0fChVk9HYDhmJ^M
8hX6+PaquiHJSVNhsv5tOshZkCfMBbyxwrKd8yZ5G5I+2gE9PUznxJaMHTmlOqm4^M
HwFxzy0RRQIDAQABo4HFMIHCMB8GA1UdIwQYMBaAFG8gWeOJIMt+aO8VuQTMzPBU^M
78k8MEoGCCsGAQUFBwEBBD4wPDA6BggrBgEFBQcwAYYuaHR0cDovL3Rlc3Q0LnJl^M
ZGJ1ZGNvbXB1dGVyLmxvY2FsOjkwODAvY2Evb2NzcDAOBgNVHQ8BAf8EBAMCBeAw^M
HQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMCQGA1UdEQQdMBuBGSRyZXF1^
M
ZXN0LnJlcXVlc3Rvcl9lbWFpbCQ=
```

同様に、OCSP には OCSP 要求情報が表示されます。

```
[07/Jul/2020:06:25:40][http-11180-Processor25]: OCSPServlet: OCSP Request:
[07/Jul/2020:06:25:40][http-11180-Processor25]: OCSPServlet:
MEUwQwIBADA+MDwwOjAJBgUrDgMCGgUABBSEWjCarLE6/BiSiENSsV9kHjqB3QQU
```

16.1.2.1. インストールログ

すべてのサブシステムはインストールログを保持します。

サブシステムが初期インストールで作成される場合や **pkispawn** によって追加のインスタンスを作成するたびに、インストールからの完全なデバッグ出力を含むインストールファイル (エラーを含む) と、インストールに成功すると、インスタンスの設定インターフェイスへの URL および PIN が作成されます。このファイルは、**/var/log/pki/** ディレクトリーに、名前が **pki-*subsystem_name*-spawn.*timestamp*.log** の形式で指定します。

インストールログの各行は、インストールプロセスのステップに従います。

例16.2 CA インストールログ

```
```

```
...
2015-07-22 20:43:13 pkispawn    : INFO     ... finalizing
'pki.server.deployment.scriptlets.finalization'
2015-07-22 20:43:13 pkispawn    : INFO     ..... cp -p /etc/sysconfig/pki/tomcat/pki-
tomcat/ca/deployment.cfg /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chmod 660 /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chown 26445:26445 /var/log/pki/pki-
tomcat/ca/archive/spawn_deployment.cfg.20150722204136
2015-07-22 20:43:13 pkispawn    : INFO     ..... generating manifest file called
'/etc/sysconfig/pki/tomcat/pki-tomcat/ca/manifest'
2015-07-22 20:43:13 pkispawn    : INFO     ..... cp -p /etc/sysconfig/pki/tomcat/pki-
tomcat/ca/manifest /var/log/pki/pki-tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chmod 660 /var/log/pki/pki-
tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : DEBUG    ..... chown 26445:26445 /var/log/pki/pki-
tomcat/ca/archive/spawn_manifest.20150722204136
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl enable pki-tomcatd.target'
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl daemon-reload'
2015-07-22 20:43:13 pkispawn    : INFO     ..... executing 'systemctl restart pki-tomcatd@pki-
tomcat.service'
2015-07-22 20:43:14 pkispawn    : INFO     END spawning subsystem 'CA' of instance 'pki-tomcat'
2015-07-22 20:43:14 pkispawn    : DEBUG
```

16.1.2.2. Tomcat のエラーとアクセスログ

CA、KRA、OCSP、TKS、および TPS サブシステムは、それらのエージェントおよびエンドエンティティーのインターフェイスに Tomcat Web サーバーインスタンスを使用します。

エラーログとアクセスログは、Certificate System とともにインストールされ、HTTP サービスを提供する Tomcat Web サーバーによって作成されます。エラーログには、サーバーが検出した HTTP エラーメッセージが含まれます。アクセスログは、HTTP インターフェイスを介したアクセスアクティビティーを一覧表示します。

Tomcat によって作成されたログ:

- admin.**timestamp**
- catalina.**timestamp**
- catalina.out
- host-manager.**timestamp**
- localhost.**timestamp**
- localhost_access_log.**timestamp**
- manager.**timestamp**

これらのログは、Certificate System 内では利用できません。それらは Apache または Tomcat 内でのみ設定可能です。これらのログの設定に関する詳細は、Apache ドキュメントを参照してください。

16.1.2.3. セルフテストログ

セルフテストのログは、サーバーの起動時またはセルフテストを手動で実行するときに取得した情報を記録します。このログを開くとテストを表示できます。このログはコンソールで設定できません。**CS.cfg** ファイルの設定を変更することでのみ設定できます。**CS.cfg** ファイルを編集してログを設定する方法は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の 『公開キューの有効化』 セクションを参照してください。

このセクションのログに関する情報は、このログには関係しません。セルフテストについての詳細は、「セルフテストの実行」を参照してください。

16.2. ログの管理

Certificate System サブシステムログファイルは、その特定のサブシステムインスタンス内の操作に関連するイベントを記録します。サブシステムごとに、インストール、アクセス、Web サーバーなどの問題について異なるログが保持されます。

すべてのサブシステムには同様のログ設定、オプション、および管理パスがあります。

16.2.1. ログ設定の概要

ログの設定方法は、Certificate System のパフォーマンスに影響を及ぼす可能性があります。たとえば、ログファイルのローテーションにより、ログが大きくなりすぎてサブシステムのパフォーマンスが低下するのを防ぎます。このセクションでは、Certificate System サブシステムによって記録されるさまざまな種類のログについて説明し、ログファイルのローテーション、バッファーリングされたログ、使用可能なログレベルなどの重要な概念を説明します。

16.2.1.1. ログに記録されるサービス

Certificate System のすべての主要コンポーネントとプロトコルは、メッセージをログファイルに記録します。表16.1「ログに記録されるサービス」 デフォルトでログに記録されるサービスを一覧表示します。特定のサービスがログに記録するメッセージを表示するには、適宜ログ設定をカスタマイズします。詳細は、「コンソールでログの表示」 を参照してください。

表16.1 ログに記録されるサービス

サービス	説明
ACL	アクセス制御リストに関連するイベントをログに記録します。
管理	コンソールとインスタンス間の HTTPS 通信など、管理アクティビティーに関連するイベントをログに記録します。
すべて	すべてのサービスに関連するイベントをログに記録します。
認証	認証モジュールに関連するアクティビティーに関連するイベントをログに記録します。
認証局	Certificate Manager に関連するイベントをログに記録します。
データベース	内部データベース関連のアクティビティーに関連するイベントをログに記録します。

サービス	説明
HTTP	サーバーの HTTP アクティビティーに関連するイベントをログに記録します。HTTP イベントは実際には、HTTP サービスを提供するために Certificate System に組み込まれる Apache サーバーに属するエラーログに記録されることに注意してください。
キーリカバリー認証局	KRA に関連するイベントをログに記録します。
LDAP	証明書と CRL の公開に使用される LDAP ディレクトリーを使用してアクティビティーに関連するイベントをログに記録します。
OCSP	OCSP ステータスの GET 要求など、OCSP に関連するイベントをログに記録します。
その他	コマンドラインユーティリティーやその他のプロセスなどの他のアクティビティーに関連するイベントをログに記録します。
要求キュー	要求キューアクティビティーに関連するイベントをログに記録します。
ユーザーおよびグループ	インスタンスのユーザーおよびグループに関連するイベントをログに記録します。

16.2.1.2. ログレベル (メッセージカテゴリー)

Certificate System サービスによってログに記録されるさまざまなイベントは、ログレベルによって決定されるため、イベントの識別とフィルタリングが簡単になります。さまざまな Certificate System のログレベルが、表16.2「ログレベルと対応するログメッセージ」 に一覧表示されます。

ログレベルは、サーバーによって実行されるログのレベルをどの程度詳細にするかを示す番号で表されます。

優先度が高いほど、優先度の高いイベントのみがログに記録されるため、詳細度が低くなります。

表16.2 ログレベルと対応するログメッセージ

ログレベル	メッセージカテゴリー	説明
0-1	トレース	これらのメッセージには、より詳細なデバッグ情報が含まれます。このレベルはパフォーマンスに影響する可能性があるため、定期的に使用しないでください。
2-5	デバッグ	これらのメッセージにはデバッグ情報が含まれます。このレベルは、非常に多くの情報を生成するため、通常の使用には推奨されません。

ログレベル	メッセージカテゴリー	説明
6-10	情報提供	これらのメッセージは、**証明書システムの初期化完了** や **成功した操作要求** などのステータスメッセージを含む、証明書システムの状態に関する一般的な情報を提供します。
11-15	警告	これらのメッセージは警告のみであり、サーバーの通常の操作に障害があることを示すものではありません。
> 15	失敗	これらのメッセージは、証明書サービス操作の実行の失敗 (**User authentication failed** または **Certificate revoked**) や、取り消せないエラーを引き起こす可能性のある予期しない状況 (**リクエストがクライアントからされた同じチャネルでクライアントに対して処理された要求を返信できない**) など、サーバーが正常に動作することを妨げるエラーおよび障害を示します。レベルを 15 より上に設定すると、障害のみが記録されるため、ログが最小限に抑えられます。

ログレベルを使用すると、イベントの重大度に基づいてログエントリーをフィルターできます。デフォルトのログレベルは 10 です。

ログデータは、特に低い (より冗長な) ログレベルでは広範囲に及ぶ可能性があります。ホストマシンには、すべてのログファイルに十分なディスク領域があることを確認します。また、すべてのログファイルがバックアップされ、ホストシステムが過負荷にならないように、ログレベル、ログローテーション、およびサーバーバックアップポリシーを適切に定義することも重要です。そうしないと、情報が失われる可能性があります。

16.2.1.3. バッファー付きおよびバッファーなしのロギング

Java サブシステムはすべてのタイプのログに対するバッファーロギングをサポートします。サーバーは、バッファー付きまたはバッファーなしのロギング用に設定できます。

バッファーログが設定されていると、サーバーは対応するログのバッファーを作成し、メッセージを可能な限りバッファーに保持します。サーバーは以下の条件のいずれかが発生した場合に限りログファイルにメッセージをフラッシュします。

- バッファーが満杯になった場合。バッファーサイズが **bufferSize** 設定パラメーターで指定された値以上になると、バッファーが満杯になります。このパラメーターのデフォルト値は 512 KB です。
- バッファーのフラッシュ間隔に到達した場合。最後のバッファーフラッシュからの経過時間、または **flushInterval** 設定パラメーターで指定された値と同じか大きい場合は、フラッシュ間隔に到達します。このパラメーターのデフォルト値は 5 秒です。
- 現在のログがコンソールから読み取られる場合。サーバーは現在のログについてクエリーされる際に最新のログを取得します。

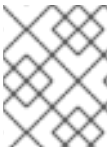
サーバーがバッファーなしロギング用に設定されている場合、サーバーはログファイルに生成されるときにメッセージをフラッシュします。サーバーはメッセージが生成されるたびに I/O 操作 (ログファイルへの書き込み) を実行するため、バッファーなしログ用にサーバーを設定するとパフォーマンスが低下します。

ログパラメーターの設定は、「コンソールでログの設定」を参照してください。

16.2.1.4. ログファイルローテーション

サブシステムログにはオプションのログ設定があり、ログファイルを無期限に拡張する代わりに、ログをローテーションして新しいログファイルを開始できます。ログファイルは、以下のいずれかの場合にローテーションされます。

- 対応するファイルのサイズ制限に到達した場合。対応するログファイルのサイズは、**maxFileSize** 設定パラメーターで指定された値以下である必要があります。このパラメーターのデフォルト値は 100 KB です。
- 対応するファイルの経過時間制限に到達した場合。対応するログファイルは、**rolloverInterval** 設定パラメーターで指定された間隔以上です。このパラメーターのデフォルト値は 2592000 秒 (30 日ごと) です。

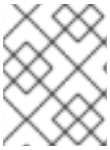


注記

これらのパラメーターを 0 に設定すると、ログファイルのローテーションが実質無効にされます。

ログファイルがローテーションされると、追加したタイムスタンプを持つファイルの名前を使用して古いファイルの名前が指定されます。追加されたタイムスタンプは、対応するアクティブなログファイルがローテーションされた日時を示す整数です。日付と時刻の形式は、YYYYMMDD (年、月、日) および HHMMSS (時、分、秒) です。

ログファイル (特に監査ログファイル)v には重要な情報が含まれています。**log** ディレクトリー全体をアーカイブメディアにコピーして、これらのファイルを定期的に一部のバックアップメディアにアーカイブする必要があります。



注記

Certificate System は、ログファイルをアーカイブするためのツールやユーティリティーを提供していません。

Certificate System は、改ざん検出の手段としてログファイルをアーカイブする前にログファイルに署名するコマンドラインユーティリティー **signtool** を提供します。詳細は、「ログファイルの署名」 を参照してください。

ログファイルの署名は、署名された監査ログ機能の代わりに使用されます。署名付き監査ログは、サブシステム署名証明書で自動的に署名される監査ログを作成します。署名済み監査ログの詳細は、「コンソールでの署名監査ログの設定」を参照してください。

ローテーションされたログファイルは削除されません。

16.2.2. コンソールでログの設定

ログは、サブシステムコンソールとサブシステムの **CS.cfg** ファイルを使用して設定できます。署名付き監査ログやカスタムログなどの特別なログは、コンソールまたは設定ファイルからも作成できます。

監査ログは、CA、OCSP、TKS、および KRA サブシステムのサブシステムコンソールで設定できます。TPS ログは、設定ファイルでのみ設定されます。

1. **Configuration** タブのナビゲーションツリーで **Log** を選択します。
2. **ログイベントリスナー管理** タブには、現在設定されているリスナーが一覧表示されます。

新しいログインスタンスを作成するには、**Add** をクリックし、**Select Log Event Listener Plug-in Implementation** ウィンドウの一覧からモジュールプラグインを選択します。

3. **Log Event Listener Editor** ウィンドウでフィールドを設定または変更します。表16.3「ログイベントリスナーフィールド」 に、さまざまなパラメーターを記載しています。

表16.3 ログイベントリスナーフィールド

フィールド	説明
Log Event Listener ID	リスナーを識別する一意の名前を指定します。この名前には、文字 (aA から zZ)、数字 (0 から 9)、アンダースコア (_)、およびハイフン (-) を使用できますが、他の文字やスペースは使用できません。
type	ログファイルのタイプを指定します。**transaction** を指定すると、監査ログが記録されます。
enabled	ログがアクティブかどうかを設定します。有効にするログのみがイベントを記録します。値は **true** または **false** です。
level	テキストフィールドにログレベルを設定します。このレベルは、フィールドに手動で入力する必要があります。選択メニューはありません。**Debug**、**Information**、**Warning**、**Failure**、**Misconfiguration**、**Catastrophe**、および **Security** を選択できます。詳細は、「ログレベル (メッセージカテゴリー)」を参照してください。
fileName	ログファイルへのファイル名を含む完全パスを指定します。サブシステムユーザーには、ファイルへの読み書きパーミッションがなければなりません。
bufferSize	ログのキロバイトサイズ (KB) のバッファーサイズを設定します。バッファーがこのサイズに達すると、バッファーの内容はフラッシュされ、ログファイルにコピーされます。デフォルトのサイズは 512 KB です。バッファーロギングの詳細は、「バッファー付きおよびバッファーなしのロギング」 を参照してください。
flushInterval	バッファーの内容がフラッシュされてログファイルに追加されるまでの時間を設定します。デフォルトの間隔は 5 秒です。
maxFileSize	ローテーションされる前に可能なログファイルのサイズをキロバイト (KB) 単位で設定できます。このサイズに達すると、ファイルはローテーションファイルにコピーされ、ログファイルが新たに開始されます。ログファイルのローテーションに関する詳細は、「ログファイルローテーション」を参照してください。デフォルトのサイズは 2000 KB です。
rolloverInterval	アクティブなログファイルをローテートするようにサーバーの頻度を設定します。利用可能なオプションは hourly、daily、weekly、monthly、および yearly です。デフォルトは monthly です。詳細は、「ログファイルローテーション」 を参照してください。

16.2.3. CS.cfg ファイルでのログの設定

CS.cfg ファイルを編集してこのログを設定する方法は、『Red Hat Certificate System の計画、インストール、およびデプロイメントのガイド』 の 『CS.cfg での CRL の更新間隔の設定』 セクションを参照してください。

16.2.4. 監査ログの管理

監査ログには、記録可能なイベントとして設定されたイベントのレコードが含まれます。**logSigning** 属性が **true** に設定されている場合、監査ログはサーバーに属するログ署名証明書で署名されます。この証明書は、ログが改ざんされていないことを確認するために監査人が使用できます。

デフォルトでは、通常の監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/**ディレクトリーと他のタイプのログにありますが、署名済み監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/signedAudit/** に書き込まれます。ログのデフォルトの場所を変更するには、設定を変更してください。

署名された監査ログは、ログ録画システムイベントを作成し、イベントが潜在的なイベント一覧から選択されます。有効にすると、署名された監査ログは、選択したイベントアクティビティーに関するメッセージの詳細セットを記録します。

署名付き監査ログは、インスタンスの初回作成時にデフォルトで設定されますが、インストール後に署名済み監査ログを設定することができます。(「インストール後の署名監査ロギングの有効化」を参照) 「コンソールでの署名監査ログの設定」 で説明されているように、設定後に設定の編集や署名証明書の変更も可能です。

16.2.4.1. 監査イベントの一覧

証明書システムの監査イベントの一覧は、付録E *監査イベント* を参照してください。

16.2.4.2. インストール後の署名監査ロギングの有効化

署名付き監査ログは、**pkispawn** コマンドに **pki_audit_group** デプロイメントパラメーターを使用して、インスタンスの初回作成時にデフォルトで有効にできます。ただし、インスタンスの作成時に署名された監査ログを設定しなかった場合には、audit ログディレクトリーの所有権を **pkiaudit** などの auditor システムユーザーグループに再割り当てすることで、このログを有効にできます。

1. インスタンスを停止します。

```
# pki-server stop instance_name
```

2. 署名付き監査ログディレクトリーのグループ所有権を、**pkiaudit** のような PKI 監査ログのオペレーティングシステムグループに設定します。これにより、PKI auditors グループのユーザーに **signedAudit** ディレクトリーへの必要な読み取りアクセスが許可され、ログファイルの署名を確認することができます。ユーザー (Certificate System ユーザーアカウント **pkiuser** を除く) に、このディレクトリーのログファイルへの書き込みアクセス権があるはずです。

```
chgrp -R pkiaudit /var/log/pki/instance_name/subsystem_name/signedAudit
```

3. インスタンスを再起動します。

```
# pki-server start instance_name
```

16.2.4.3. コンソールでの署名監査ログの設定

署名付き監査ログは、インスタンスの初回作成時にデフォルトで設定されますが、設定後に設定を編集するか、署名証明書を変更することが可能です。

注記

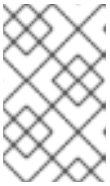
署名された監査ログは大きくなる可能性があるため、ファイルシステムに十分なスペースを確保してください。

logSigning パラメーターを **enable** に設定し、ログの署名に使用される証明書のニックネームを指定することにより、ログが署名済み監査ログに設定されます。特別なログ署名証明書は、サブシステムの初回設定時に作成されます。

auditor 権限を持つユーザーのみが、署名済み監査ログにアクセスでき、表示できます。監査担当者は、**AuditVerify** ツールを使用して、署名済み監査ログが改ざんされていないことを確認できます。

署名付き監査ログはサブシステムの設定時に作成され、有効になりますが、監査ログの作成および署名を行うには追加の設定が必要になります。

1. コンソールを開きます。

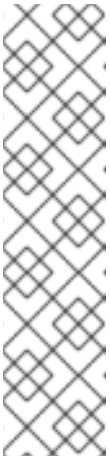


注記

CS.cfg ファイルを編集して監査ログを作成または設定するには、『Red Hat Certificate System 計画、インストール、およびデプロイメントガイド』 の 『CS.cfg ファイルのログの設定』 を参照してください。

2. **Configuration** タブのナビゲーションツリーで **Log** を選択します。
3. **ログイベントリスナー管理** タブで、**SignedAudit** エントリーを選択します。
4. **Edit/View** をクリックします。
5. **Log Event Listener Editor** ウィンドウでリセットする必要のある 3 つのフィールドがあります。

- **signedAuditCertNickname** を入力します。これは、監査ログの署名に使用される証明書のニックネームです。監査署名証明書はサブシステムが設定されているときに作成され、**auditSigningCert cert-*instance_name subsystem_name*** のようなニックネームがあります。



注記

監査署名証明書のニックネームを取得するには、**certutil** を使用してサブシステムの証明書データベースの証明書を一覧表示します。以下に例を示します。

```
certutil -L -d /var/lib/pki-tomcat/alias

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-pki-tomcat            u,u,u
Server-Cert cert-pki-tomcat              u,u,u
auditSigningCert cert-pki-tomcat CA      u,u,Pu
```

- **logSigning** フィールドを **true** に設定して、署名済みロギングを有効にします。

- 監査ログに記録される **イベント** を設定します。付録E *監査イベント* ログ可能なイベントを一覧表示します。ログイベントは、空白のないコンマで区切ります。

6. ファイル名、ログレベル、ファイルサイズ、ローテーションスケジュールなど、ログに関するその他の設定を行います。

注記

デフォルトでは、通常の監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/** ディレクトリーと他のタイプのログにありますが、署名済み監査ログは **/var/log/pki/*instance_name*/*subsystem_name*/signedAudit/** に書き込まれます。ログのデフォルトの場所を変更するには、設定を変更してください。

7. ログ設定を保存します。

署名付き監査ログを有効にした後、ユーザーを作成し、そのエントリーを監査人グループに割り当てることにより、監査人ユーザーを割り当てます。監査グループのメンバーは、署名された監査ログを表示して検証できる唯一のユーザーです。auditors の設定に関する詳細は、「ユーザーの作成」 を参照してください。

監査担当者は、**AuditVerify** ツールを使用してログを確認できます。このツールの使用方法は、man ページの **AuditVerify(1)** を参照してください。

16.2.4.4. 監査ロギングエラーの処理

監査ロギング機能が失敗する可能性があるイベントがあるため、イベントをログに書き込むことができません。たとえば、監査ログファイルが含まれるファイルシステムが満杯であったり、ログファイルのファイル権限が誤って変更されると、監査ログのロギングが失敗する可能性があります。監査ロギングが失敗すると、Certificate System インスタンスは以下のようにシャットダウンします。

- サーブレットが無効になり、新しいリクエストを処理しません。
- 保留中のリクエストと新しいリクエストはすべて強制終了されます。
- サブシステムがシャットダウンしています。

これが発生すると、管理者と監査人はオペレーティングシステム管理者と協力して、ディスク領域またはファイルパーティションの問題を解決する必要があります。IT の問題が解決したら、監査人は最後の監査ログエントリーが署名されていることを確認する必要があります。そうでない場合は、今後監査検証の失敗を防ぐために、手動で署名 (「ログファイルの署名」)、アーカイブし、削除する必要があります。これが完了すると、管理者は Certificate System を再起動することができます。

16.2.4.5. ログファイルの署名

Certificate System は、監査目的でアーカイブまたは配布される前にログファイルへのデジタル署名を行うことができます。この機能により、ファイルの改ざんを確認できます。

これは、署名された監査ログ機能の代替機能です。署名付き監査ログ機能は、自動的に署名される監査ログを作成します。このツールは、アーカイブされたログを手動で署名します。署名済み監査ログの詳細は、「コンソールでの署名監査ログの設定」を参照してください。

ログファイルの署名には、署名ツール (**signtool**) と呼ばれるコマンドラインユーティリティーを使用します。このユーティリティーの詳細は、http://www.mozilla.org/projects/security/pki/nss/tools/ を参照してください。

ユーティリティーは、サブシステムインスタンスの証明書、キー、およびセキュリティーモジュールデータベースの情報を使用します。

auditor 権限を持つユーザーとしては、**signtool** コマンドを使用してログディレクトリーに署名します。

```
signtool -d secdb_dir -k cert_nickname -Z output input
```

- **secdb_dir** は、CA の証明書、キー、およびセキュリティーモジュールデータベースが含まれるディレクトリーへのパスを指定します。
- **cert_nickname** は、署名に使用する証明書のニックネームを指定します。
- **output** は JAR ファイルの名前 (署名された zip ファイル) を指定します。
- **input** は、ログファイルを含むディレクトリーへのパスを指定します。

16.2.4.6. 監査イベントのフィルタリング

Certificate System では、管理者はフィルターを設定して、イベント属性に基づいて監査ファイルに記録される監査イベントを設定できます。

フィルターの形式は LDAP フィルターと同じです。ただし、Certificate System は、以下のフィルターのみをサポートします。

表16.4 サポート対象の Audit イベントフィルター

タイプ	形式	例
Presence	(*attribute*=*)	(ReqID=*)
Equality	(*attribute=value*)	(Outcome=Failure)
Substring	(*attribute=initial*any*...*any*final*)	(SubjectID=*admin*)
AND 演算	(&(*filter_1*)(*filter_2*)...(*filter_n*))	(&(SubjectID=admin)(Outcome=Failure))
OR 演算	(|(*filter_1*)(*filter_2*)...(*filter_n*))	(|(SubjectID=admin)(Outcome=Failure))
NOT 演算	(!(*filter*))	(!(SubjectID=admin))

LDAP フィルターの詳細は、『Red Hat Directory Server 管理ガイド』の『複合検索フィルター』を参照してください。

例16.3 監査イベントのフィルタリング

InfoName フィールドが **rejectReadon** または **cancelReason** に設定されている処理済み証明書要求のイベントと、プロファイル証明書要求およびイベントの失敗したイベントのみを記録するには、以下を実行します。

1. **/var/lib/pki/*instance_name*/*subsystem_type*/conf/CS.cfg** ファイルを編集して、以下のパラメーターを設定します。

```
log.instance.SignedAudit.filters.PROFILE_CERT_REQUEST=(Outcome=Failure)
log.instance.SignedAudit.filters.CERT_REQUEST_PROCESSED=(|
(InfoName=rejectReason)(InfoName=cancelReason))
```

2. Certificate System を再起動します。

```
# pki-server restart instance_name
```

16.2.5. ログモジュールの管理

許可されるログのタイプとそれらの動作は、**ログモジュール** プラグインで設定されます。新しいロギングモジュールが作成され、カスタムログの作成に使用できます。

新しいログプラグインモジュールは、コンソールから登録できます。新しいモジュールを登録するには、モジュール名と、ログインターフェイスを実装する Java™ クラスのフルネームを指定する必要があります。

プラグインモジュールを登録する前に、モジュール用の Java™ クラスを **classes** ディレクトリーに配置します。実装はクラスパス上になければなりません。

ログプラグインモジュールをサブシステムインスタンスで登録するには、以下を実行します。

1. カスタムジョブクラスを作成します。この例では、カスタムログプラグインは **MyLog.java** と呼ばれます。
2. インスタンスの lib ディレクトリーに新しいクラスをコンパイルします。

```
javac -d . /var/lib/pki/pki-tomcat/lib -classpath $CLASSPATH MyLog.java
```

3. CA がカスタムクラスにアクセスできるように、CA の **WEB-INF** Web ディレクトリーにディレクトリーを作成します。

```
mkdir /var/lib/pki/pki-tomcat/webapps/ca/WEB-INF/classes
```

4. 所有者を Certificate System のシステムユーザー (**pkiuser**) に設定します。

```
chown -R pkiuser:pkiuser /var/lib/pki/pki-tomcat/lib
```

5. プラグインを登録します。
   a. コンソールにログインします。
   b. **Configuration** タブで、ナビゲーションツリーから **Logs** を選択します。次に、**Log Event Listener Plug-in Registration** タブを選択します。
   c. **Register** をクリックします。

      Register Log Event Listener Plug-in Implementation ウィンドウが表示されます。
   d. プラグインモジュールの名前と、Java™ クラス名を指定します。

      Java™ クラス名は、実装する Java™ クラスの完全パスです。このクラスがパッケージに含

まれる場合は、パッケージ名を含めます。たとえば、**com.customplugins** という名前のパッケージに **customLog** という名前のクラスを登録すると、クラス名は **com.customplugins.customLog** になります。

e. **OK** をクリックします。

不要なログプラグインモジュールは、コンソールから削除できます。モジュールを削除する前に、このモジュールに基づくリスナーをすべて削除します。「ログファイルローテーション」を参照してください。

16.3. ログの使用

16.3.1. コンソールでログの表示

サブシステムのトラブルシューティングを行うには、サーバーがログ記録したエラーまたは情報メッセージを確認します。ログファイルを調べると、サーバーの操作の多くの側面も監視できます。一部のログファイルは、コンソールで表示できます。ただし、監査ログには、「署名監査ログの使用」 で説明している方法を使用して、Auditor ロールを持つユーザーのみがアクセスできます。

ログファイルの内容を表示するには、次の手順を実行します。

1. コンソールにログインします。
2. **Status** タブを選択します。
3. **Logs** で表示するログを選択します。
4. **Display Options** セクションで表示設定を行います。
   - **Entries**: 表示するエントリーの最大数。この制限に達すると、Certificate System は検索要求に一致するエントリーを返します。ゼロ (0) はメッセージが返されないことを意味します。フィールドが空の場合、サーバーは見つかった数に関係なく、一致するすべてのエントリーを返します。
   - **Source**: ログメッセージが表示される証明書システムコンポーネントまたはサービスを選択します。**All** を選択すると、このファイルにログ記録するすべてのコンポーネントによってログに記録されるメッセージが表示されます。
   - **レベル**: メッセージのフィルターに使用するログレベルを表すメッセージカテゴリーを選択します。
   - **ファイル名**: 表示するログファイルを選択します。
5. **Refresh** をクリックします。
6. 完全なエントリーを表示するには、エントリーをダブルクリックしてそのエントリーを選択し、**View** をクリックします。

16.3.2. 署名監査ログの使用

このセクションでは、署名された監査ログを Auditor グループのユーザーが表示および検証する方法を説明します。

16.3.2.1. 監査ログの一覧表示

auditor 権限を持つユーザーは、**pki *subsystem*-audit-file-find** コマンドを使用して、サーバー上の既存の監査ログファイルを一覧表示します。

たとえば、**server.example.com** にホストされる CA の監査ログファイルを一覧表示するには、次のコマンドを実行します。

```
# pki -h server.example.com -p 8443 -n auditor ca-audit-file-find
-----------------
3 entries matched
-----------------
 File name: ca_audit.20170331225716
 Size: 2883

 File name: ca_audit.20170401001030
 Size: 189

 File name: ca_audit
 Size: 6705
---------------------------
Number of entries returned 3
---------------------------
```

このコマンドは、CA に対して認証するために、**auditor** ディレクトリーに保存されている *auditor* ニックネームのあるクライアント証明書を使用します。コマンドで使用するパラメーターおよび代替の認証方法の詳細は、man ページの pki(1) を参照してください。

16.3.2.2. 監査ログのダウンロード

auditor 権限を持つユーザーとして、**pki *subsystem*-audit-file-retrieve** コマンドを使用して、サーバーから特定の監査ログをダウンロードします。

たとえば、**server.example.com** でホストされる CA から監査ログファイルをダウンロードするには、以下を実行します。

1. 任意で、CA で利用可能なログファイルを一覧表示します。「監査ログの一覧表示」を参照してください。
2. ログファイルをダウンロードします。たとえば、**ca_audit** ファイルをダウンロードします。

   ```
   # pki -U https://server.example.com:8443 -n auditor ca-audit-file-retrieve ca_audit
   ```

   このコマンドは、CA に対して認証するために、**auditor** ディレクトリーに保存されている *auditor* ニックネームのあるクライアント証明書を使用します。コマンドで使用するパラメーターおよび代替の認証方法の詳細は、man ページの pki(1) を参照してください。

ログファイルをダウンロードした後、**grep** ユーティリティーを使用して、特定のログエントリーを検索できます。

```
# grep "\[AuditEvent=ACCESS_SESSION_ESTABLISH\]" log_file
```

16.3.2.3. 署名済み監査ログの確認

監査ログの署名が有効な場合、監査権限を持つユーザーはログを確認することができます。

1. NSS データベースを初期化し、CA 証明書をインポートします。詳細は、「pki CLI の初期化」および『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『NSS データベースへの証明書のインポート』セクションを参照してください。

2. 監査署名証明書が PKI クライアントデータベースにない場合は、インポートします。

   a. 確認するサブシステムログについて監査署名証明書を検索します。以下に例を示します。

```
# pki ca-cert-find --name "CA Audit Signing Certificate"
---------------
1 entries found
---------------
 Serial Number: 0x5
 Subject DN: CN=CA Audit Signing Certificate,O=EXAMPLE
 Status: VALID
 Type: X.509 version 3
 Key Algorithm: PKCS #1 RSA with 2048-bit key
 Not Valid Before: Fri Jul 08 03:56:08 CEST 2016
 Not Valid After: Thu Jun 28 03:56:08 CEST 2018
 Issued On: Fri Jul 08 03:56:08 CEST 2016
 Issued By: system
----------------------------
Number of entries returned 1
----------------------------
```

   b. 監査署名証明書を PKI クライアントにインポートします。

```
# pki client-cert-import "CA Audit Signing Certificate" --serial 0x5 --trust ",,P"
-----------------------------------------------------
Imported certificate "CA Audit Signing Certificate"
-----------------------------------------------------
```

3. 監査ログをダウンロードします。「監査ログのダウンロード」を参照してください。

4. 監査ログを確認します。

   a. 検証する監査ログファイルのリストを時系列で含むテキストファイルを作成します。以下に例を示します。

```
# cat > ~/audit.txt << EOF
ca_audit.20170331225716
ca_audit.20170401001030
ca_audit
EOF
```

   b. **AuditVerify** ユーティリティーを使用して署名を確認します。以下に例を示します。

```
# AuditVerify -d ~/.dogtag/nssdb/ -n "CA Audit Signing Certificate" \
   -a ~/audit.txt
Verification process complete.
Valid signatures: 10
Invalid signatures: 0
```

   AuditVerify の使用方法は、AuditVerify(1) man ページを参照してください。

16.3.3. オペレーティングシステムレベルの監査ログの表示

注記

以下の手順を使用してオペレーティングシステムレベルの監査ログを表示するには、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の 『OS レベルの監査ログの有効化』 セクションに従って **auditd** ロギングフレームワークを設定する必要があります。

オペレーティングシステムレベルのアクセスログを表示するには、root として **ausearch** ユーティリティーを使用するか、**sudo** ユーティリティーを使用して特権ユーザーとして使用します。

16.3.3.1. 監査ログ削除イベントの表示

これらのイベントは、(**rhcs_audit_deletion** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_deletion
```

16.3.3.2. シークレットおよび秘密鍵の NSS データベースへのアクセスの表示

これらのイベントは、(**rhcs_audit_nssdb** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_nssdb
```

16.3.3.3. 時間変更イベントの表示

これらのイベントはキー化されるため (**rhcs_audit_time_change**を使用)、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_time_change
```

16.3.3.4. パッケージ更新イベントの表示

これらのイベントは、(**SOFTWARE_UPDATE** タイプの) タイプ付きメッセージであるため、***-m*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -m SOFTWARE_UPDATE
```

16.3.3.5. PKI 設定変更の表示

これらのイベントは、(**rhcs_audit_config** を使用して) キーが設定されているため、***-k*** パラメーターを使用して、そのキーに一致するイベントを検索します。

```
# ausearch -k rhcs_audit_config
```

16.3.4. スマートカードのエラーコード

スマートカードは、TPS に特定のエラーコードを報告できます。これは、メッセージの原因に応じて TPS のデバッグログファイルに記録されます。

表16.5 スマートカードのエラーコード

戻りコード	説明
一般的なエラーコード	
6400	特定の診断なし
6700	Lc の誤った長さ
6982	セキュリティーステータスが満たされない
6985	使用条件が満たされない
6a86	間違った P1 P2
6d00	無効な命令
6e00	無効なクラス
インストール読み込みエラー	
6581	メモリー障害
6a80	データフィールドの誤ったパラメーター
6a84	不十分なメモリー容量
6a88	参照データが見つからない
削除エラー	
6200	アプリケーションを論理的に削除
6581	メモリー障害
6985	参照データを削除できない
6a88	参照データが見つからない
6a82	アプリケーションが見つからない
6a80	コマンドデータの値が正しくない
データ取得エラー	

戻りコード	説明
6a88	参照データが見つからない
ステータス取得エラー	
6310	より多くのデータが利用可能
6a88	参照データが見つからない
6a80	コマンドデータの値が正しくない
読み込みエラー	
6581	メモリー障害
6a84	不十分なメモリー容量
6a86	間違った P1/P2
6985	使用条件が満たされない

第17章 サブシステム証明書の管理

本章では、証明書の使用の概要を示します。使用できるタイプと形式、HTML エンドエンティティーフォームと Certificate System コンソールを使用して証明書を要求および作成する方法、Certificate System とさまざまなクライアントに証明書をインストールする方法です。さらに、コンソールを介した証明書の管理と、証明書を使用するためのサーバー設定に関する情報があります。

17.1. 必要なサブシステム証明書

各サブシステムには、操作を実行するためにサブシステムインスタンスに発行する必要のある証明書の定義されたセットがあります。Certificate Manager の設定中に設定される証明書の内容の特定の詳細があり、証明書のタイプに応じて制約、設定、および属性に関するさまざまな考慮事項があります。証明書のフォーマットの計画については、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 を参照してください。

17.1.1. 証明書マネージャー証明書

Certificate Manager がインストールされると、CA 署名証明書、SSL サーバー証明書、および OCSP 署名証明書の鍵および要求が生成されます。この証明書は、設定を完了する前に作成されます。

CA 証明書要求は、CA への自己署名リクエストとして送信されます。次に、証明書を発行して自己署名ルート CA の作成を終了するか、サードパーティーのパブリック CA または別の Certificate System CA に送信されます。外部 CA が証明書を返すと、証明書がインストールされ、下位 CA のインストールが完了します。

- 「CA 署名キーペアおよび証明書」
- 「OCSP 署名キーペアおよび証明書」
- 「サブシステム証明書」
- 「SSL サーバーキーペアおよび証明書」
- 「Audit ログ署名キーペアおよび証明書」

17.1.1.1. CA 署名キーペアおよび証明書

すべての Certificate Manager には、Certificate Manager が発行する証明書と CRL に署名するために使用する秘密鍵に対応する公開鍵を持つ CA 署名証明書があります。この証明書は、Certificate Manager のインストール時に作成され、インストールされます。証明書のデフォルトのニックネームは **caSigningCert cert-instance_ID CA** です。ここで、**instance_ID** は Certificate Manager インスタンスを識別します。証明書のデフォルトの有効期間は 5 年間です。

CA 署名証明書のサブジェクト名は、インストール時に設定された CA の名前を反映します。Certificate Manager によって署名または発行されたすべての証明書には、証明書の発行者を識別するためにこの名前が含まれています。

Certificate Manager のステータスがルートまたは下位 CA として評価されるかどうかは、その CA 署名証明書が自己署名の証明書であるか、または別の CA により署名されているかにより決まります。これは、証明書のサブジェクト名に影響します。

- Certificate Manager がルート CA の場合、その CA 署名証明書は自己署名の証明書です。つまり、証明書のサブジェクト名と発行者名は同じです。
- Certificate Manager が下位 CA である場合、その CA 署名証明書は別の CA、通常は CA 階層の

上位レベル (ルート CA である場合とそうでない場合があります) によって署名されます。Certificate Manager を使用して証明書を発行する前に、ルート CA の署名証明書を個別のクライアントおよびサーバーにインポートする必要があります。

注記

CA 名を変更 **できない** か、以前に発行された証明書がすべて無効化されている。同様に、新しい鍵ペアで CA 署名証明書を再発行し、古い鍵ペアで署名された証明書をすべて無効にします。

17.1.1.2. OCSP 署名キーペアおよび証明書

OCSP 署名証明書のサブジェクト名は **cn=OCSP cert-instance_ID CA** の形式であり、OCSP レスポンスの署名に必要な拡張機能 (**OCSPSigning** および **OCSPNoCheck** など) が含まれます。

OCSP 署名証明書のデフォルトのニックネームは **ocspSigningCert cert-instance_ID** で、**instance_ID CA** は証明書マネージャーインスタンスを識別します。

OCSP 署名証明書の公開鍵に対応する OCSP 秘密鍵は、証明書失効リストのステータスをクエリーするときに Certificate Manager が OCSP 準拠のクライアントに署名するために使用されます。

17.1.1.3. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー SSL 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。セキュリティードメイン CA 自体の場合は、そのサブシステム証明書自体によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-instance_ID** です。

17.1.1.4. SSL サーバーキーペアおよび証明書

すべての証明書マネージャーには、証明書マネージャーのインストール時に最初に生成された SSL サーバー証明書が少なくとも 1 つ含まれます。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。**instance_ID** は証明書マネージャーインスタンスを識別します。

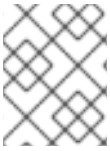
デフォルトでは、認証に Certificate Manager は SSL サーバー証明書を使用します。ただし、エンドエンティティーサービスインターフェイスとエージェントサービスインターフェイスへの認証に個別のサーバー証明書を使用するように証明書マネージャーを設定するなど、さまざまな操作に使用する追加のサーバー証明書を要求できます。

Certificate Manager が公開ディレクトリーとの SSL 対応通信用に設定されている場合、デフォルトではクライアント認証に SSL サーバー証明書を使用します。証明書マネージャーは、SSL クライアント認証に別の証明書を使用するように設定することもできます。

17.1.1.5. Audit ログ署名キーペアおよび証明書

CA は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。



注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

17.1.2. オンライン証明書ステータスマネージャーの証明書

Online Certificate Status Manager が最初に設定されていると、必要な証明書の鍵がすべて作成され、OCSP 署名、SSL サーバー、監査ログ署名、およびサブシステム証明書の証明書要求が作成されます。これらの証明書要求は CA (Certificate System CA またはサードパーティー CA のいずれか) に送信され、設定プロセスを完了するには Online Certificate Status Manager データベースにインストールする必要があります。

- 「SSL サーバーキーペアおよび証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」
- 「オンライン証明書ステータスマネージャー証明書の認識」

17.1.2.1. OCSP 署名キーペアおよび証明書

すべての Online Certificate Status Manager には、証明書である OCSP 署名証明書があります。これには、Online Certificate Status Manager が OCSP 応答に署名するために使用する秘密鍵に対応する公開鍵があります。Online Certificate Status Manager の署名は、Online Certificate Status Manager がリクエストを処理している永続的な証明を提供します。この証明書は、Online Certificate Status Manager が設定されている場合に生成されます。証明書のデフォルトのニックネームは **ocspSigningCert cert-instance_ID** で、**instance_ID OSCP** は Online Certificate Status Manager インスタンス名です。

17.1.2.2. SSL サーバーキーペアおよび証明書

すべての Online Certificate Status Manager には、Online Certificate Status Manager の設定時に生成された SSL サーバー証明書が少なくとも 1 つ含まれます。証明書のデフォルトのニックネームは **Server-Cert cert-instance_ID** です。ここで、**instance_ID** は Online Certificate Status Manager インスタンス名を識別します。

Online Certificate Status Manager は、Online Certificate Status Manager エージェントサービスページでサーバー側の認証にサーバー証明書を使用します。

Online Certificate Status Manager は認証目的で単一のサーバー証明書を使用します。追加のサーバー証明書をインストールして、さまざまな目的で使用できます。

17.1.2.3. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー SSL 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-instance_ID** です。

17.1.2.4. Audit ログ署名キーペアおよび証明書

OCSP は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

注記

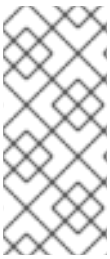
その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

17.1.2.5. オンライン証明書ステータスマネージャー証明書の認識

Online Certificate Status Manager の SSL サーバー証明書に署名した CA によっては、Certificate Manager が認識する証明書および発行する CA を取得しないといけない場合があります。

- Online Certificate Status Manager のサーバー証明書が CRL を公開している CA によって署名されている場合は、何もする必要はありません。
- Online Certificate Status Manager のサーバー証明書が、下位の Certificate Manager の証明書に署名したのと同じルート CA によって署名されている場合、ルート CA は、下位の Certificate Manager の証明書データベースで信頼できる CA としてマークする必要があります。
- Online Certificate Status Manager の SSL サーバー証明書が別のルート CA によって署名されている場合は、ルート CA 証明書を下位の Certificate Manager の証明書データベースにインポートし、信頼できる CA としてマークする必要があります。

Online Certificate Status Manager のサーバー証明書が、選択したセキュリティードメインの CA によって署名されている場合は、Online Certificate Status Manager の設定時に証明書チェーンがインポートされ、マークされます。他の設定は必要ありません。ただし、サーバー証明書が外部 CA で署名されている場合は、設定を完了するために証明書チェーンをインポートする必要があります。



注記

セキュリティードメイン内のすべての CA が、設定時に OCSP Manager によって自動的に信頼されるわけではありません。ただし、CA パネルで設定された CA の証明書チェーン内のすべての CA は、OCSP Manager によって自動的に信頼されます。セキュリティードメイン内にあるが証明書チェーンにはない他の CA は、手動で追加する必要があります。

17.1.3. キーリカバリー認証局の証明書

KRA は、以下のキーペアと証明書を使用します。

- 「トランスポートキーペアおよび証明書」
- 「ストレージキーペア」
- 「SSL サーバー証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

17.1.3.1. トランスポートキーペアおよび証明書

すべての KRA にはトランスポート証明書があります。トランスポート証明書の生成に使用されるキーペアの公開キーは、アーカイブのために KRA に送信される前に、エンドエンティティーの秘密暗号化

キーを暗号化するためにクライアントソフトウェアによって使用されます。デュアルキーペアを生成できるクライアントのみがトランスポート証明書を使用します。

17.1.3.2. ストレージキーペア

すべての KRA にはストレージキーペアがあります。 KRA は、このキーペアの公開コンポーネントを使用して、キーをアーカイブするときに秘密暗号化キーを暗号化 (またはラップ) します。プライベートコンポーネントを使用して、リカバリー中にアーカイブされたキーを復号化 (またはアンラップ) します。このキーペアの使用方法に関する詳細は、*4章 キーアーカイブおよびリカバリーの設定* を参照してください。

ストレージキーで暗号化したキーは、承認されたキーリカバリーエージェントによりのみ取得できます。

17.1.3.3. SSL サーバー証明書

すべての Certificate System KRA には、少なくとも 1 つの SSL サーバー証明書があります。最初の SSL サーバー証明書は、KRA が設定される際に生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-**instance_ID です。**instance_id** は KRA インスタンスを識別します。

KRA の SSL サーバー証明書は、証明書要求が送信される CA により発行されました。これは Certificate System CA またはサードパーティー CA です。発行者名を表示するには、KRA コンソールの **System Keys and Certificates** オプションで証明書の詳細を開きます。

KRA は、KRA エージェントサービスインターフェイスに対するサーバー側の認証に SSL サーバー証明書を使用します。デフォルトでは、Key Recovery Authority は認証に単一の SSL サーバー証明書を使用します。ただし、追加の SSL サーバー証明書を要求し、KRA にインストールすることができます。

17.1.3.4. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー SSL 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-**instance_ID です。

17.1.3.5. Audit ログ署名キーペアおよび証明書

KRA は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

17.1.4. TKS 証明書

TKS には 3 つの証明書があります。SSL サーバーおよびサブシステムの証明書が標準の操作に使用されます。監査ログの保護には、追加の署名証明書が使用されます。

- 「SSL サーバー証明書」

- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

17.1.4.1. SSL サーバー証明書

すべての Certificate System TKS には、少なくとも 1 つの SSL サーバー証明書があります。最初の SSL サーバー証明書は、TKS が設定される際に生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-**instance_ID です。

17.1.4.2. サブシステム証明書

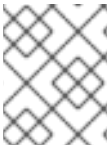
セキュリティードメインのすべてのメンバーには、サーバー SSL 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-**instance_ID です。

17.1.4.3. Audit ログ署名キーペアおよび証明書

TKS は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。



注記

その他の証明書は ECC キーを使用できますが、監査署名証明書は **常に** RSA キーを使用する必要があります。

17.1.5. TPS 証明書

TPS は、サーバー証明書、サブシステム証明書、監査ログ署名証明書の 3 つの証明書のみを使用します。

- 「SSL サーバー証明書」
- 「サブシステム証明書」
- 「Audit ログ署名キーペアおよび証明書」

17.1.5.1. SSL サーバー証明書

すべての Certificate System TPS には、少なくとも 1 つの SSL サーバー証明書があります。TPS が設定されると、最初の SSL サーバー証明書が生成されます。証明書のデフォルトのニックネームは **Server-Cert cert-**instance_ID です。

17.1.5.2. サブシステム証明書

セキュリティードメインのすべてのメンバーには、サーバー SSL 証明書とは別のドメインメンバー間の通信に使用するサーバー証明書が発行されます。この証明書はセキュリティードメイン CA によって署名されます。

証明書のデフォルトのニックネームは **subsystemCert cert-**instance_ID です。

17.1.5.3. Audit ログ署名キーペアおよび証明書

TPS は、サーバーで発生したすべてのイベントのセキュアな監査ログを保持します。監査ログが改ざんされていないことを保証するために、ログファイルは特別なログ署名証明書によって署名されます。

監査ログ署名証明書は、サーバーの初回設定時に発行されます。

17.1.6. サブシステム証明書のキータイプについて

新規インスタンスの作成時には、**pkispawn** ユーティリティーに渡される設定ファイルでキーのタイプとキーサイズを指定できます。

例17.1 CA のキータイプ関連の設定パラメーター

以下は、例の値を含む主要なタイプ関連のパラメーターです。これらのパラメーターは、新規 CA の作成時に **pkispawn** に渡される設定ファイルで設定できます。

```
pki_ocsp_signing_key_algorithm=SHA256withRSA
pki_ocsp_signing_key_size=2048
pki_ocsp_signing_key_type=rsa

pki_ca_signing_key_algorithm=SHA256withRSA
pki_ca_signing_key_size=2048
pki_ca_signing_key_type=rsa

pki_sslserver_key_algorithm=SHA256withRSA
pki_sslserver_key_size=2048
pki_sslserver_key_type=rsa

pki_subsystem_key_algorithm=SHA256withRSA
pki_subsystem_key_size=2048
pki_subsystem_key_type=rsa

pki_admin_keysize=2048
pki_admin_key_size=2048
pki_admin_key_type=rsa

pki_audit_signing_key_algorithm=SHA256withRSA
pki_audit_signing_key_size=2048
pki_audit_signing_key_type=rsa
```

注記

サンプルの値は CA 用です。他のサブシステムには異なるパラメーターが必要です。

詳細は、以下を参照してください。

- 『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『ユーティリティーの理解』セクション
- パラメーターおよび例の説明の pki_default.cfg(5) man ページ。

17.1.7. HSM を使用したサブシステム証明書の保存

デフォルトでは、鍵と証明書は、**/var/lib/pki/*instance_name*/alias** ディレクトリーのローカル管理のデータベースである **key4.db** および **cert9.db** に保存されます。ただし、Red Hat Certificate System は、ハードウェアセキュリティーモジュール (HSM) もサポートします。この外部デバイスは、ネットワーク上にある集中的な場所に鍵と証明書を保存できます。HSM を使用すると、インスタンスのキーと証明書に簡単にアクセスできるため、クローン作成などの一部の機能が簡単になります。

HSM を使用して証明書を保存する場合、HSM 名は証明書のニックネームに付加され、**server.xml** ファイルなどのサブシステム設定にフルネームが使用されます。以下に例を示します。

```
serverCert="nethsm:Server-Cert cert-instance_ID
```

注記

1 つの HSM を使用して、複数のホストにインストールする可能性がある複数のサブシステムインスタンスの証明書および鍵を保存することができます。HSM を使用する場合、サブシステムの証明書ニックネームは HSM で管理される各サブシステムインスタンスに対して一意である必要があります。

証明書システムは、nCipher netHSM と Chrysalis LunaSA の 2 種類の HSM に対応します。

17.2. コンソールを使用した証明書の要求

CA、OCSP、KRA、および TKS の Certificate Setup Wizard は、サブシステム証明書の証明書登録プロセスを自動化します。コンソールは、そのサブシステムによって使用される証明書の要求および証明書を作成、送信、およびインストールできます。これらの証明書は、サーバー証明書、または CA 署名証明書や KRA トランスポート証明書などのサブシステム固有の証明書にすることができます。

17.2.1. 署名証明書の要求

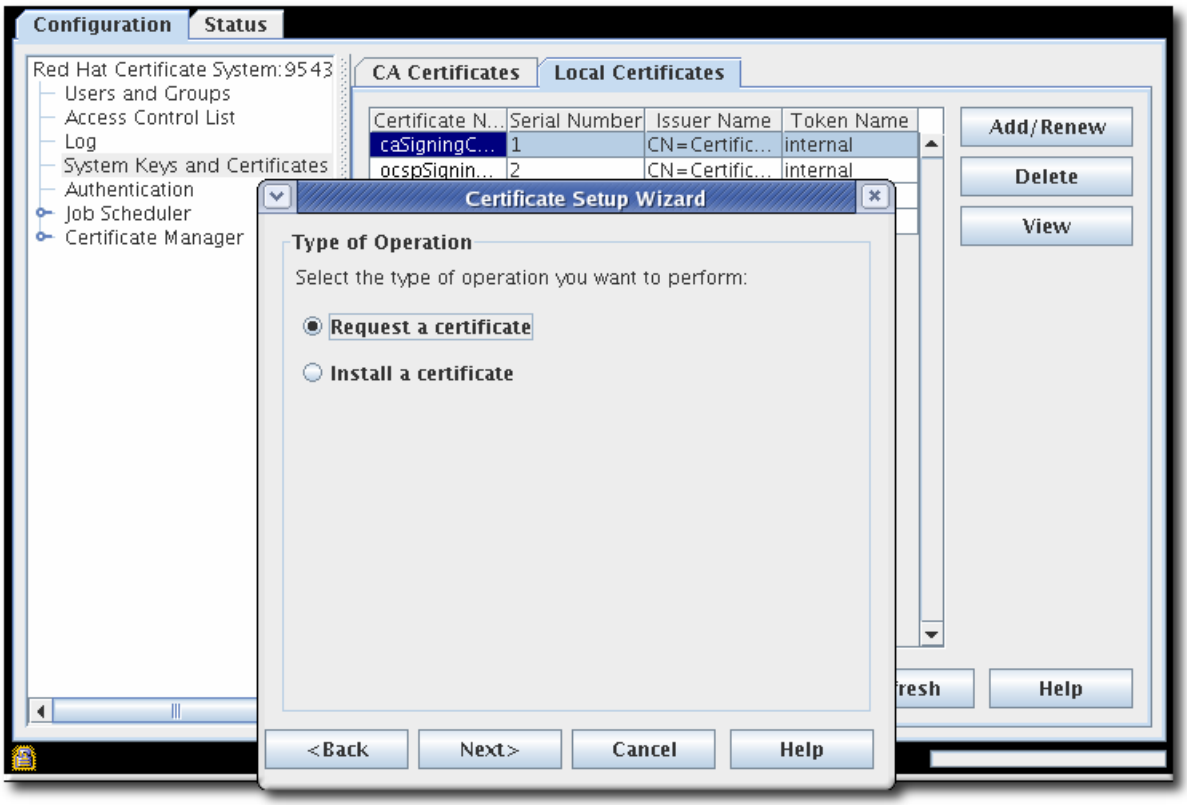
注記

ユーザーがサブシステムにアクセスするために使用するコンピューターからクライアント要求を生成および送信することが重要です。これは、要求プロセスの一部がローカルマシンに秘密鍵を生成するためです。場所独立性が必要な場合には、スマートカードなどのハードウェアトークンを使用してキーペアと証明書を保存します。

1. サブシステムコンソールを開きます。以下に例を示します。

   ```
   pkiconsole https://server.example.com:8443/ca
   ```

2. **Configuration** タブで、ナビゲーションツリーで **System Keys and Certificates** を選択します。
3. 右側のパネルで **Local Certificates** タブを選択します。
4. **Add/Renew** をクリックします。



5. **証明書要求** ラジオボタンを選択します。

6. 要求の署名証明書タイプを選択します。

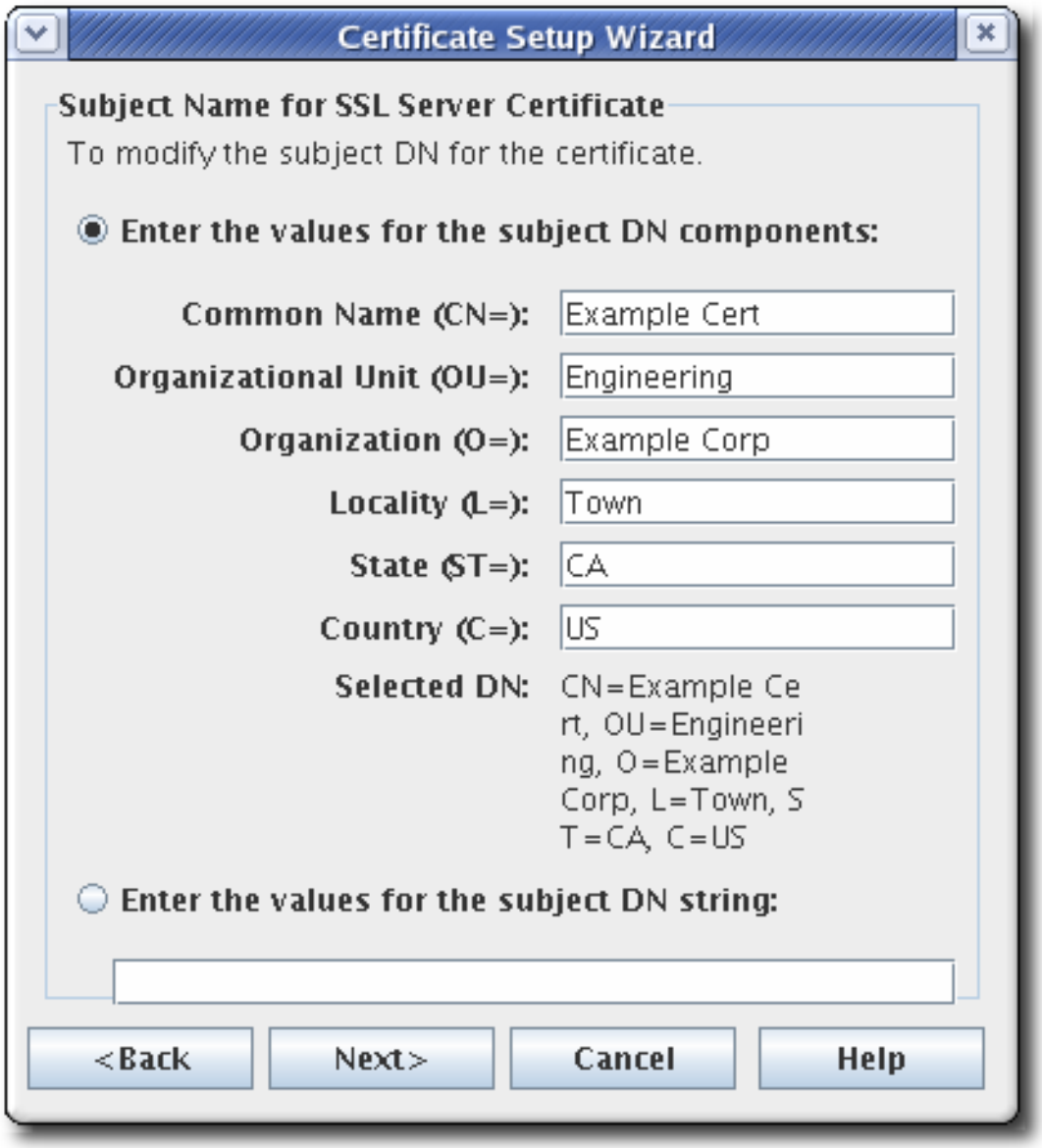
7. ルート CA または下位 CA のいずれかの CA のタイプを選択します。

8. キーペアの情報を設定し、キー (トークン) を生成する場所を設定します。これは内部セキュリティーデータベースディレクトリーまたは一覧表示された外部トークンのいずれかになります。

   新しい証明書を作成するには、新しいキーペアを作成する必要があります。既存のキーペアを使用すると、既存の証明書が更新されるだけです。

9. メッセージダイジェストアルゴリズムを選択します。

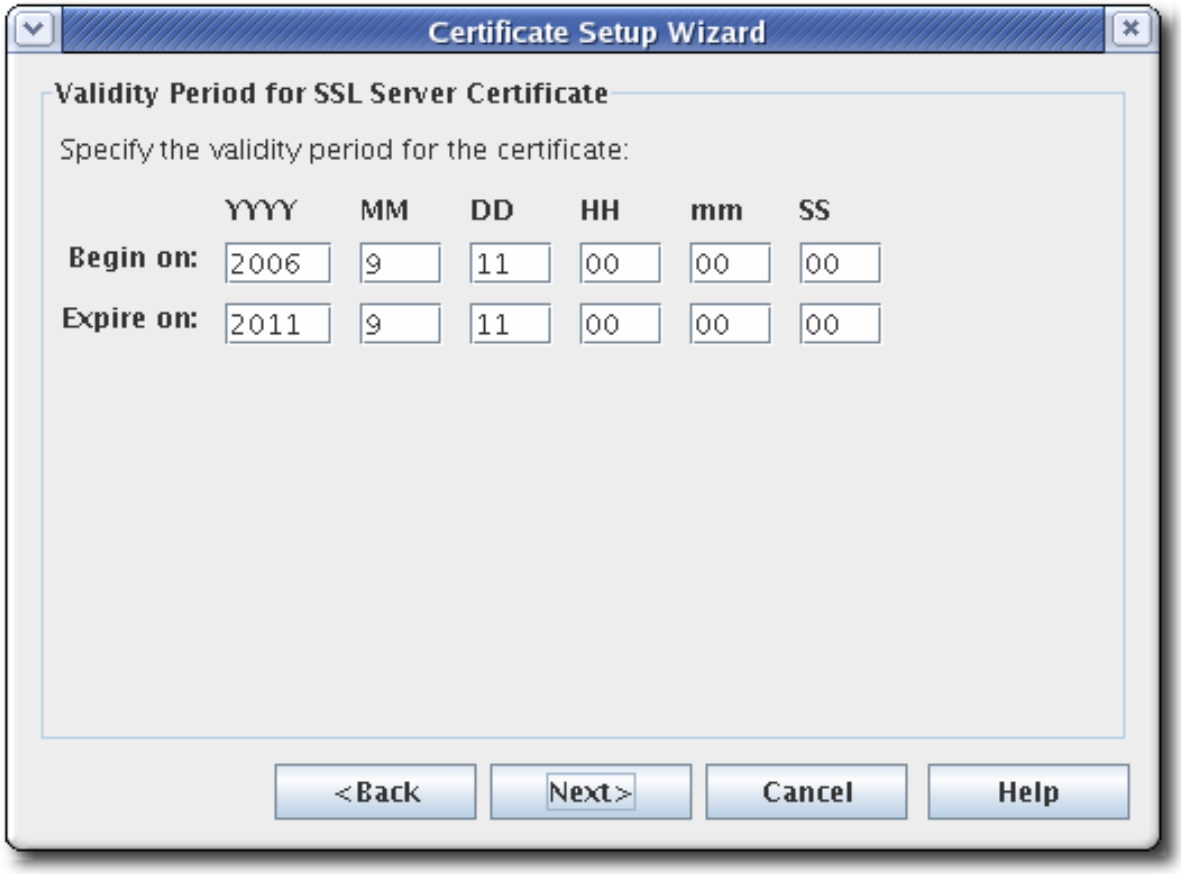
10. サブジェクト名を指定します。サブジェクト DN を構築するための個別の DN 属性の値を入力するか、完全な文字列を入力します。



証明書要求フォームは、共通名、組織単位、および要求側の名前フィールドの UTF-8 文字をすべてサポートします。

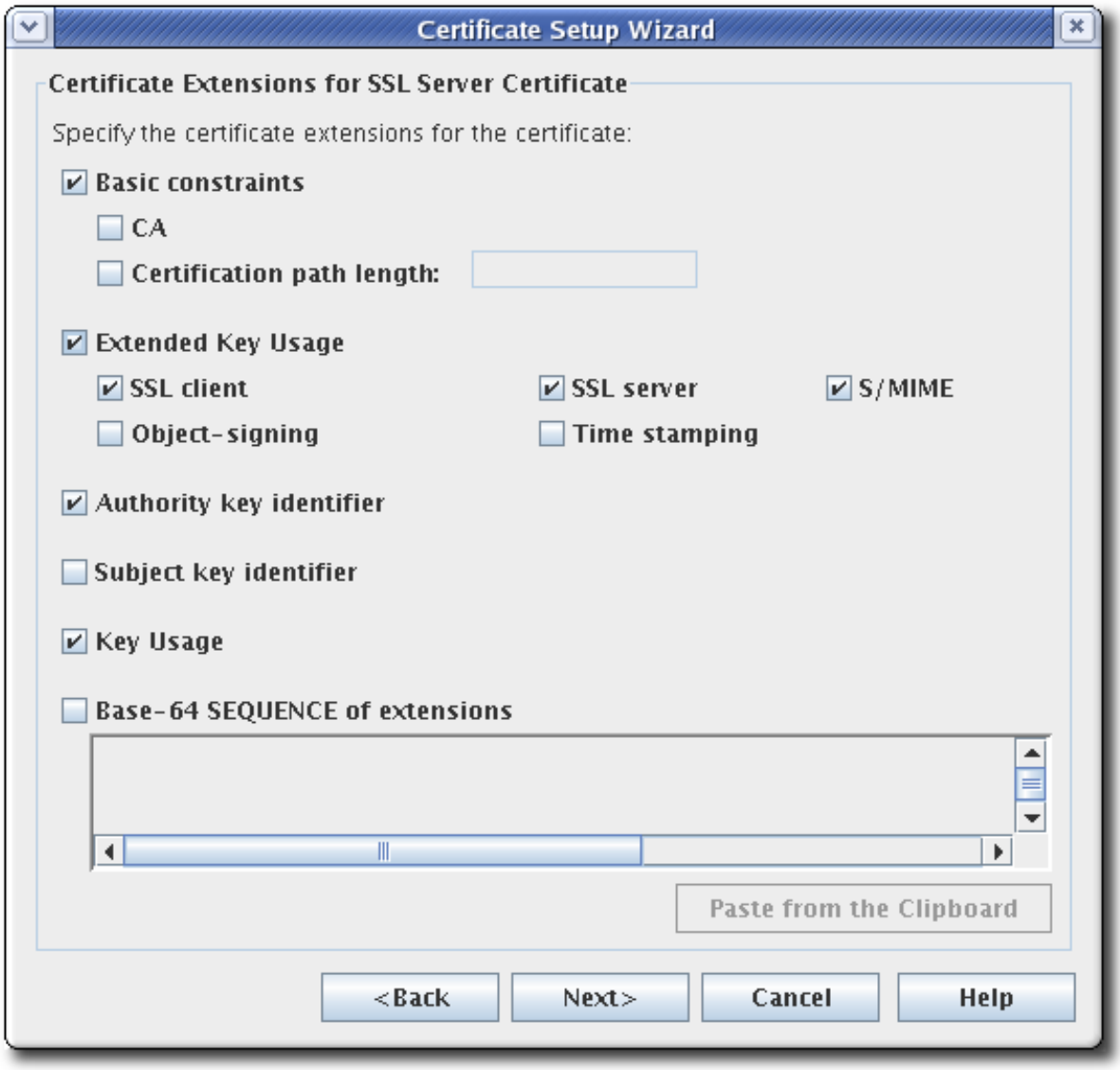
このサポートには、国際化されたドメイン名のサポートは含まれません。

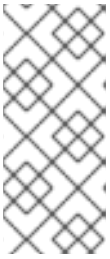
11. 証明書の有効期間の開始日と終了日、およびそれらの日付で有効期間が開始および終了する時刻を指定します。



デフォルトの有効期間は 5 年間です。

12. 証明書の標準拡張機能を設定します。必要な拡張機能はデフォルトで選択されます。デフォルトの選択を変更するには、付録B *証明書およびCRL のデフォルト、制約、および拡張* で説明されているガイドラインを参照してください。





注記

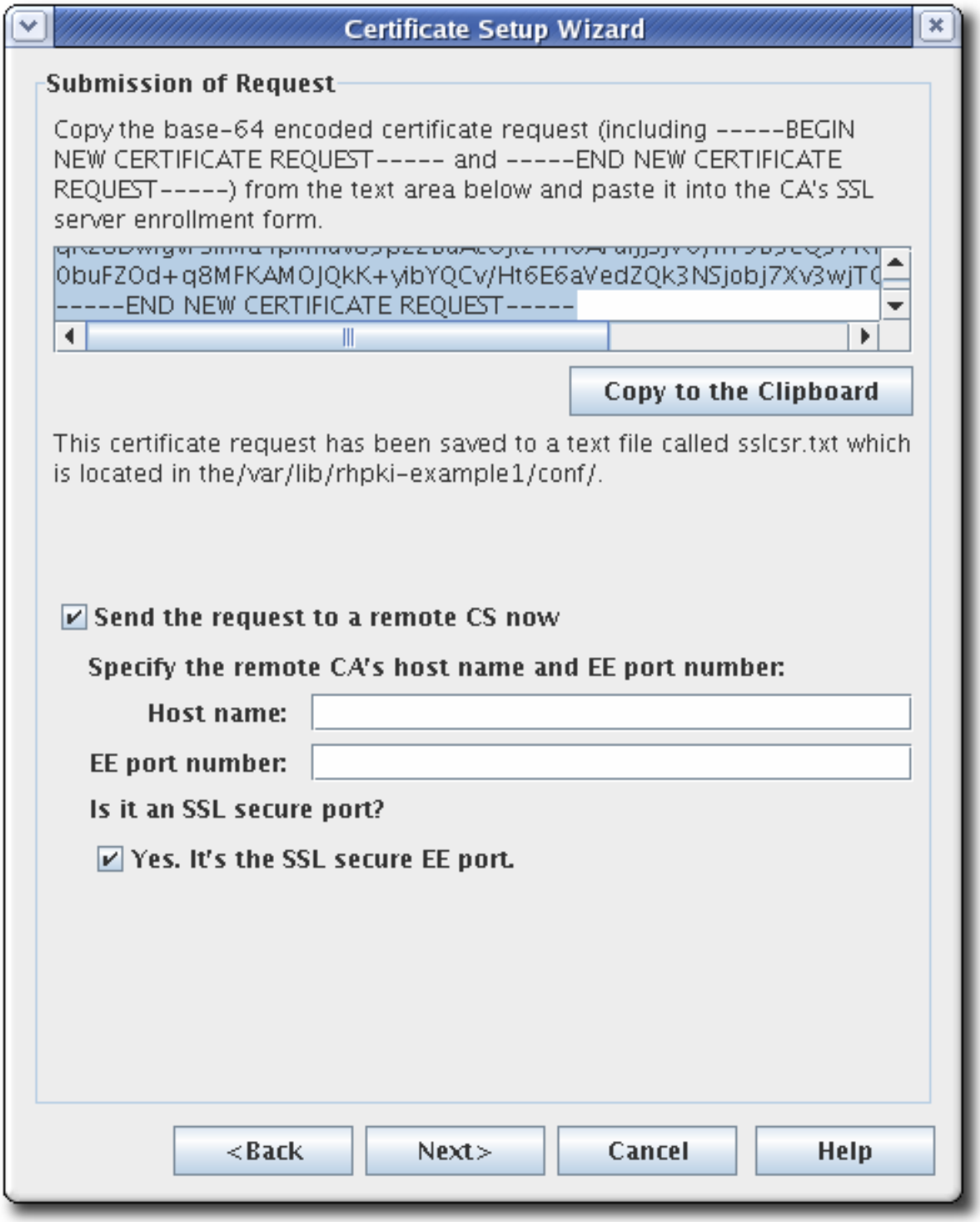
CA 階層の設定には、証明書拡張が必要です。下位 CA には、下位 SSL CA (SSL の証明書を発行できるようにする) または下位電子メール CA (安全な電子メールの証明書を発行できるようにする) のいずれかとして識別する拡張子を含む証明書が必要です。証明書拡張を無効にすると、CA 階層が設定できないことを意味します。

- 基本の制約。関連付けられたフィールドは CA 設定であり、証明書パスの長さの数値設定になります。
- 拡張鍵の使用。
- 認証局キー識別子。
- サブジェクトキー識別子。
- 鍵の使用方法。デフォルトでは、デジタル署名 (ビット 0)、否認防止 (ビット 1)、キー証明書サイン (ビット 5)、および CRL サイン (ビット 6) ビットが設定されています。拡張機能は、PKIX 標準および RFC 2459 によって推奨されているとマークされます。Key Usage 拡張機能の説明は、RFC 2459 を参照してください。

- 拡張機能の Base-64 SEQUENCE。これはカスタム拡張用です。MIME 64 DER でエンコードされた形式のエクステンションをテキストフィールドに貼り付けます。

複数の拡張機能を追加するには、**ExtJoiner** プログラムを使用します。ツールの使用方法は、『Certificate System コマンドラインツールガイド』を参照してください。

13. ウィザードはキーペアを生成し、証明書署名要求を表示します。



リクエストは base-64 でエンコードされた PKCS #10 形式であり、マーカーファイル **-----BEGIN NEW CERTIFICATE REQUEST-----** および **-----END NEW CERTIFICATE REQUEST-----** でバインドされます。以下に例を示します。

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXRzY2F
```

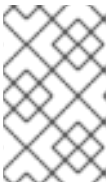
```
wZSBDb21tdW5pY2
F0aW9uczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAwMFo
XDTk5MDIyMzE5MDA
wMnbjdgngYoxIDAeBgNVBAoTF05ldHNjYXBlIENvbW11bmljYXRpb25zMQ8wDQYDVQQLEw
ZQZW9wbGUxFz
AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjMCEGC
SqGSIb3Dbndg
JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4QgEB
AQHBAQDAgCAM
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hIBgdR+jWeL
mD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

ウィザードは、証明書要求を、**/var/lib/pki/*instance_name*/*subsystem_type*/conf/** にある設定ディレクトリーに作成したテキストファイルにコピーします。テキストファイルの名前は、要求された証明書の種類によって異なります。設定可能なテキストファイルは 表17.1「証明書署名要求用に作成されたファイル」 に記載されています。

表17.1 証明書署名要求用に作成されたファイル

ファイル名	証明書署名要求
cacsr.txt	CA 署名証明書
ocspcsr.txt	証明書マネージャーの OCSP 署名証明書
ocspcsr.txt	OCSP 署名証明書

CA に送信する前に、証明書要求を変更しないでください。リクエストはウィザード経由で自動的に送信することも、クリップボードにコピーして、終了ページを介して CA に手動で送信することもできます。



注記

ウィザードの自動送信機能は、リモートの Certificate Manager にのみ要求を送信できます。サードパーティーの CA に要求を送信するのに使用できません。サードパーティーの CA に送信するには、証明書要求ファイルを使用します。

14. 証明書を取得します。

    a. Certificate Manager エンドエンティティーを開きます。

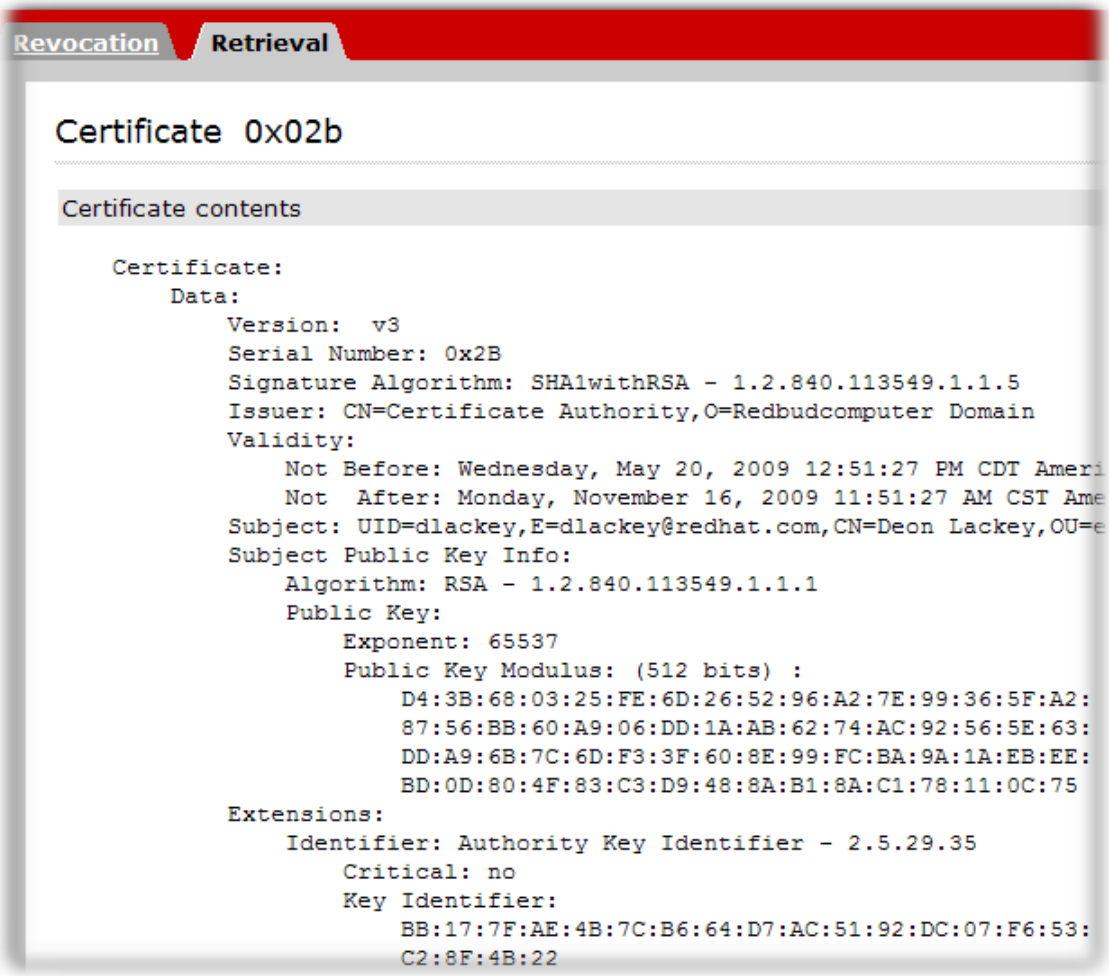
    ```
    https://server.example.com:8443/ca/ee/ca
    ```

    b. **Retrieval** タブをクリックします。

    c. 証明書要求の送信時に作成された要求 ID 番号を入力し、**Submit** をクリックします。

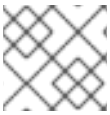
    d. 次のページには、証明書要求のステータスが表示されます。ステータスが **complete** すると、証明書へのリンクがあります。**Issued certificate** のリンクをクリックします。

e. 新しい証明書情報は、pretty-print 形式、base-64 エンコード形式、および PKCS #7 形式で表示されます。



f. base-64 でエンコードされた証明書 (マーカー行 **-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** を含む) をテキストファイルにコピーします。テキストファイル

を保存し、それを使用して証明書のコピーをサブシステムの内部データベースに保存します。「ユーザーの作成」を参照してください。



注記

pkiconsole が非推奨になりました。

17.2.2. 他の証明書の要求

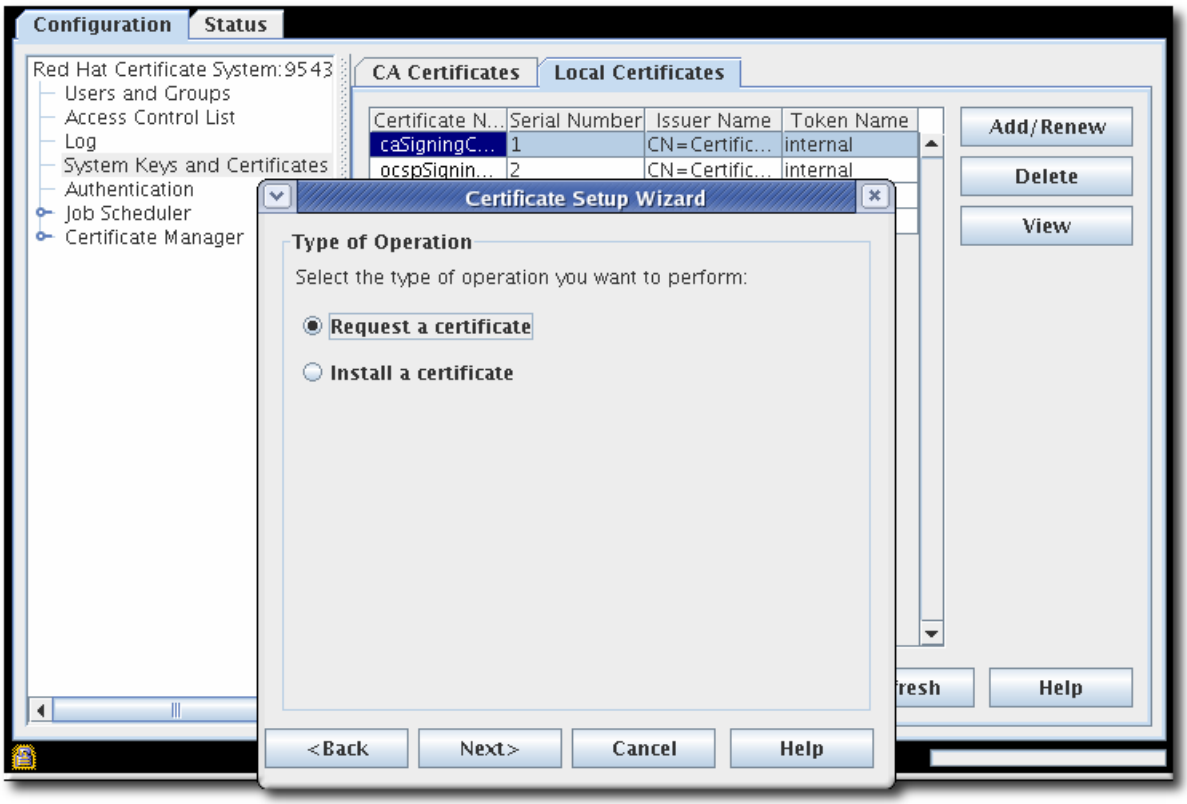
注記

ユーザーがサブシステムにアクセスするために使用するコンピューターからクライアント要求を生成および送信することが重要です。これは、要求プロセスの一部がローカルマシンに秘密鍵を生成するためです。場所独立性が必要な場合には、スマートカードなどのハードウェアトークンを使用してキーペアと証明書を保存します。

1. サブシステムコンソールを開きます。以下に例を示します。

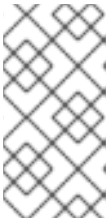
```
pkiconsole https://server.example.com:8443/ca
```

2. **Configuration** タブで、ナビゲーションツリーで **System Keys and Certificates** を選択します。
3. 右側のパネルで **Local Certificates** タブを選択します。
4. **Add/Renew** をクリックします。



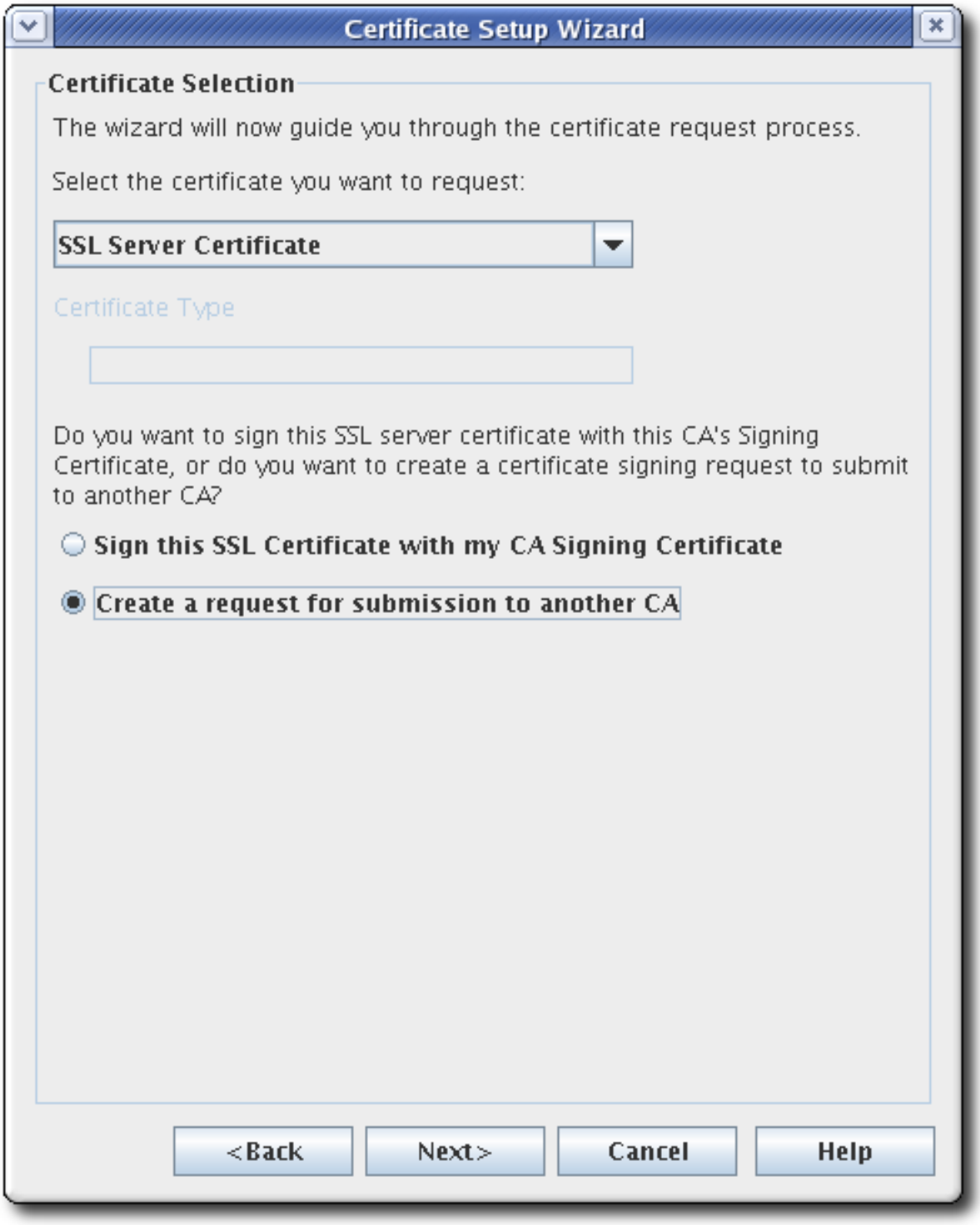
5. **証明書要求** ラジオボタンを選択します。

6. 要求する証明書のタイプを選択します。要求できる証明書のタイプはサブシステムによって異なります。



注記

その他の証明書を作成することを選択すると、**Certificate Type** フィールドが有効になります。作成する証明書のタイプを入力します。CRL 署名証明書の **caCrlSigning**、監査ログ署名証明書の **caSignedLogCert**、または SSL クライアント証明書用の **client** を入力します。

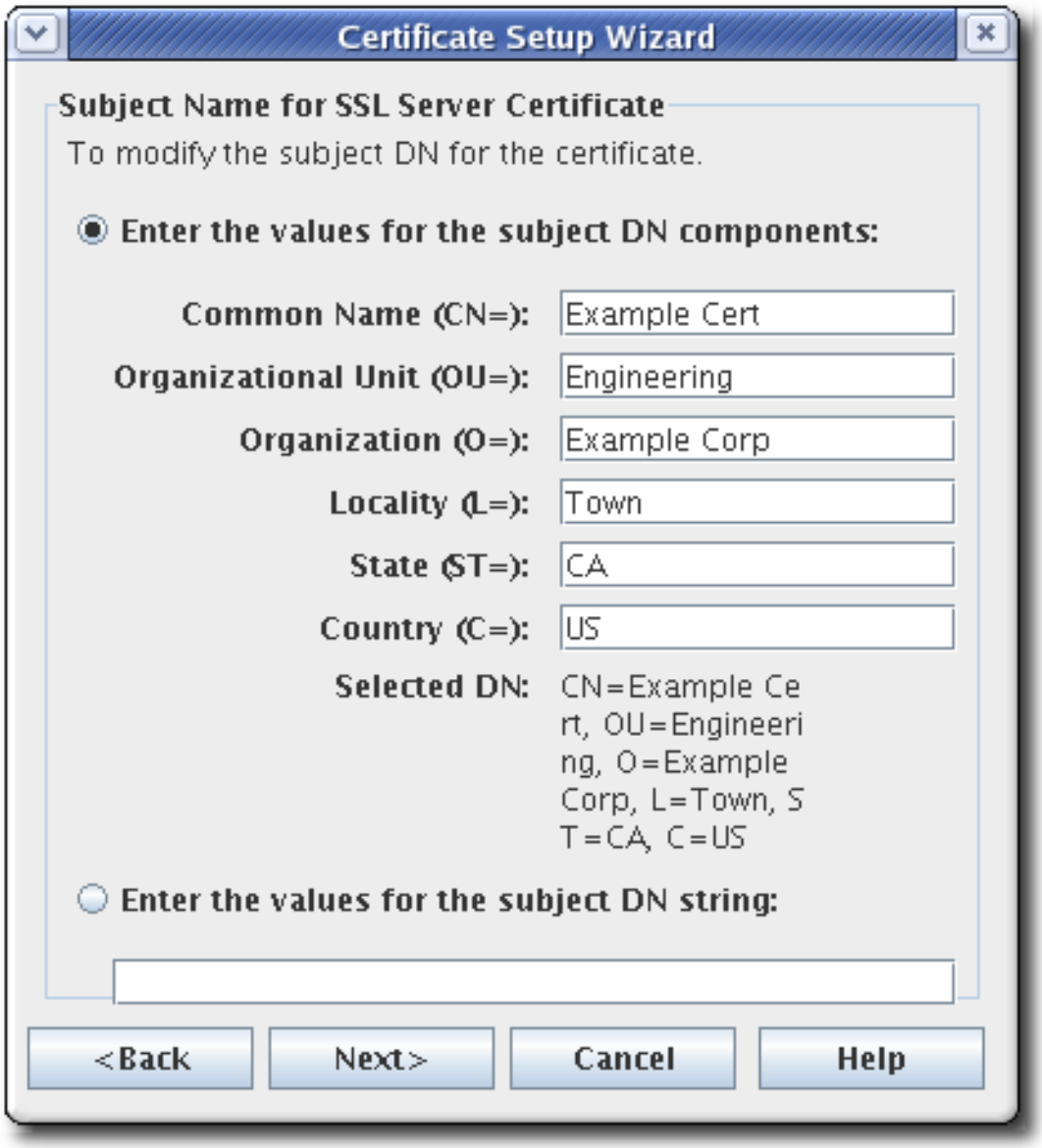


7. 要求に署名する CA のタイプを選択します。このオプションは、ローカルの CA 署名証明書を使用するか、別の CA に送信するリクエストを作成します。

8. キーペアの情報を設定し、キー (トークン) を生成する場所を設定します。これは内部セキュリティーデータベースディレクトリーまたは一覧表示された外部トークンのいずれかになります。

   新しい証明書を作成するには、新しいキーペアを作成する必要があります。既存のキーペアを使用すると、既存の証明書が更新されるだけです。

9. サブジェクト名を指定します。サブジェクト DN を構築するための個別の DN 属性の値を入力するか、完全な文字列を入力します。



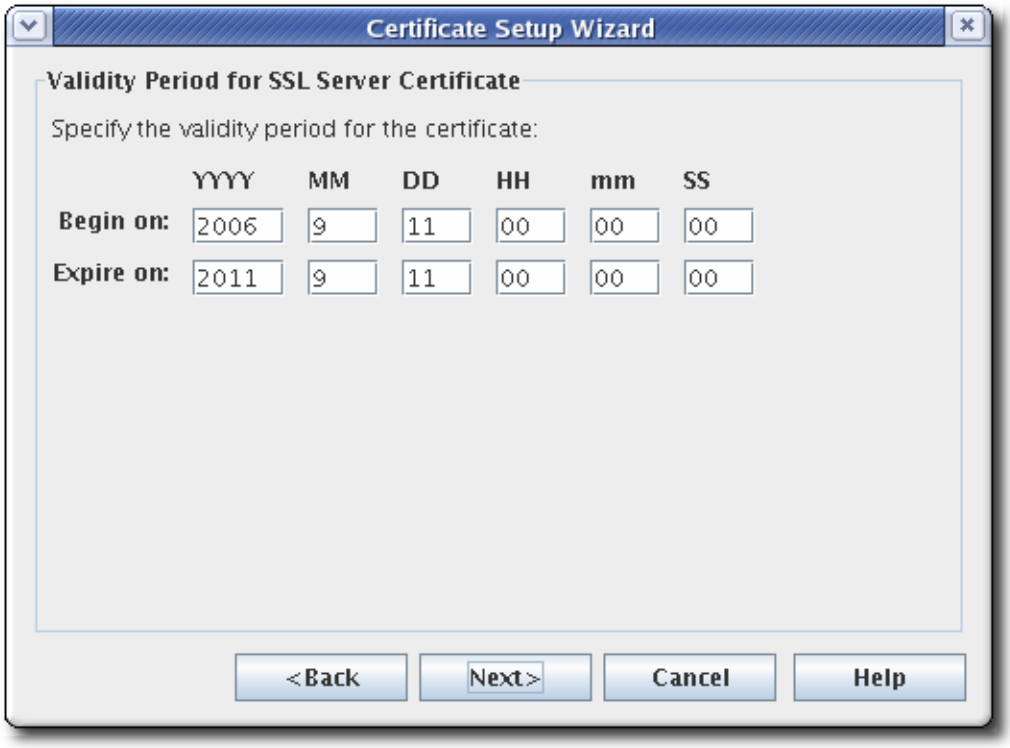
注記

SSL サーバー証明書の場合、共通名は **machine_name.domain.domain** 形式の Certificate System の完全修飾ホスト名である必要があります。

CA 証明書要求フォームは、共通名、組織単位、および要求側の名前フィールドの UTF-8 文字をすべてサポートします。

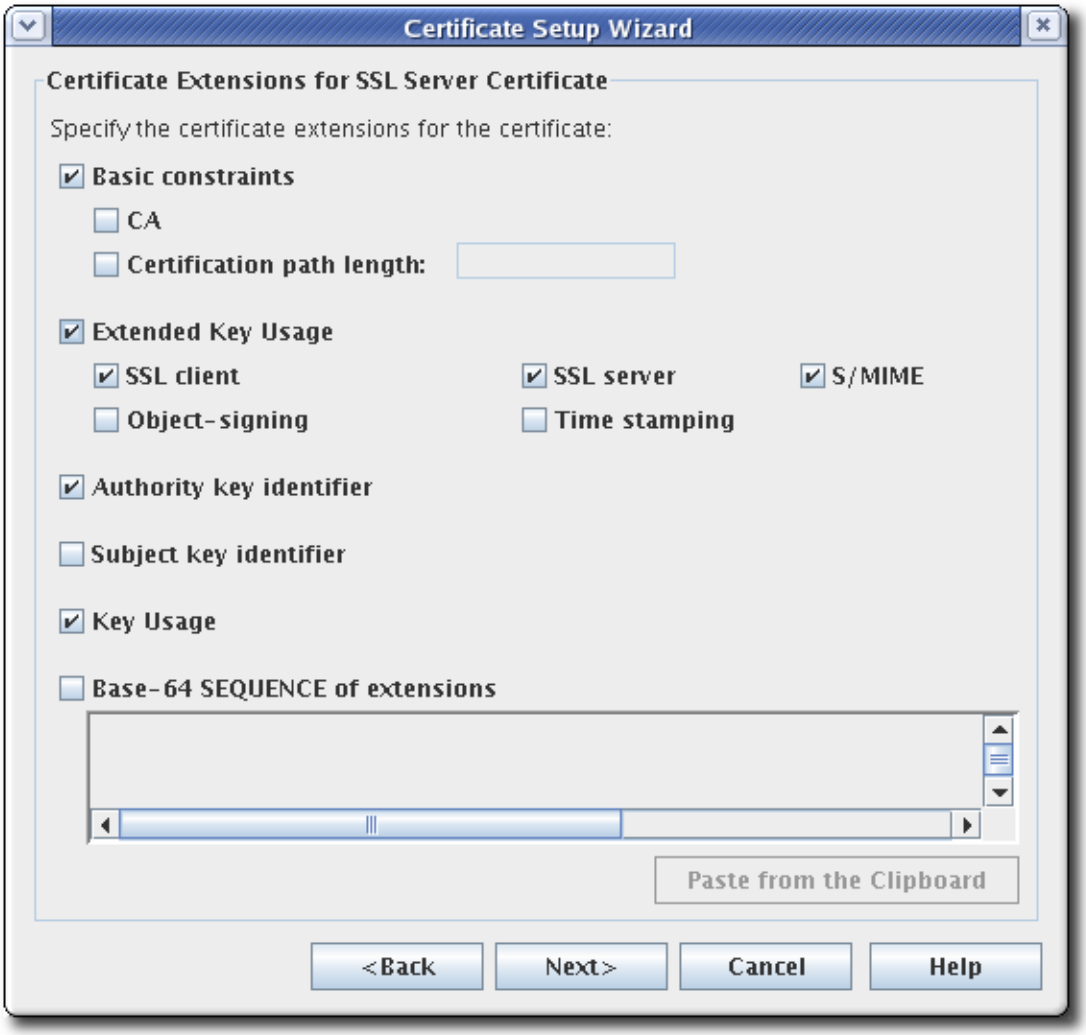
このサポートには、国際化されたドメイン名のサポートは含まれません。

10. 証明書の有効期間の開始日と終了日、およびそれらの日付で有効期間が開始および終了する時刻を指定します。



デフォルトの有効期間は 5 年間です。

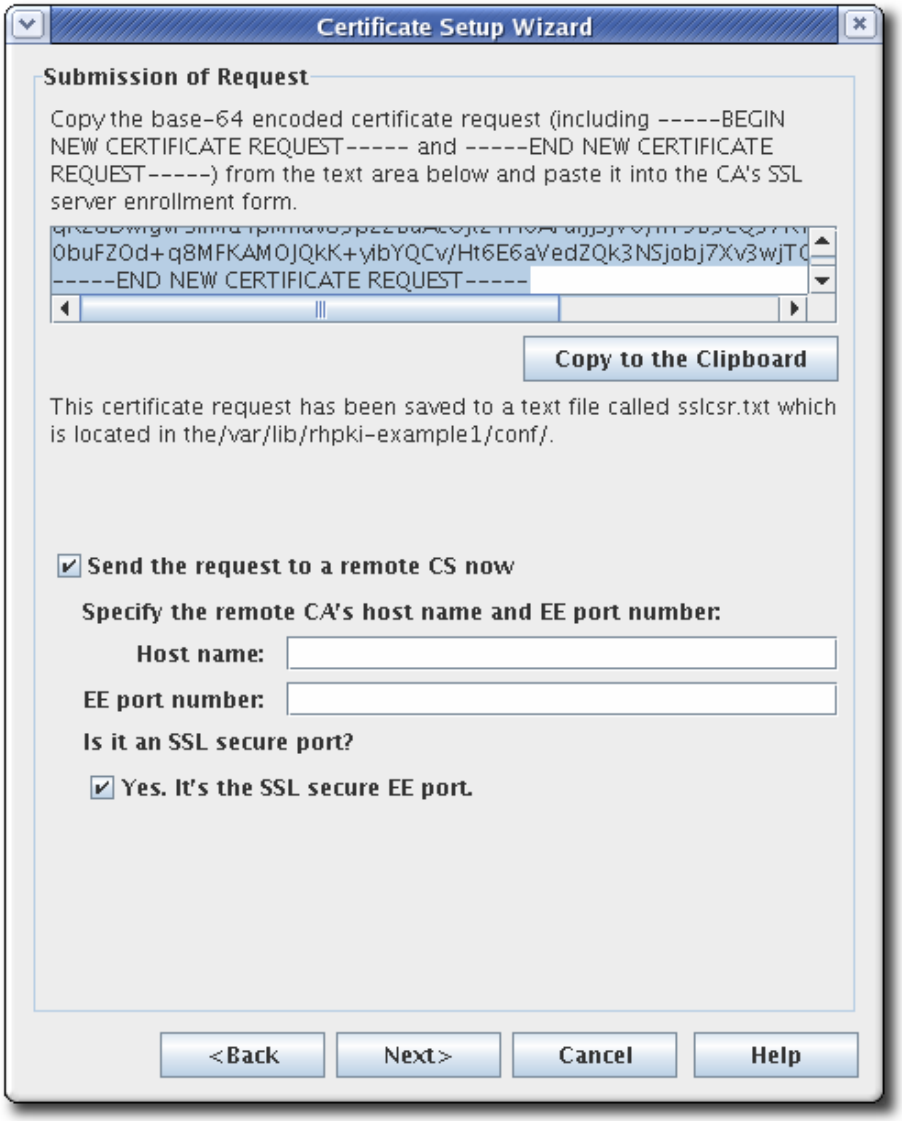
11. 証明書の標準拡張機能を設定します。必要な拡張機能はデフォルトで選択されます。デフォルトの選択を変更するには、付録B *証明書および CRL のデフォルト、制約、および拡張* で説明されているガイドラインを参照してください。



- 拡張鍵の使用。
- 認証局キー識別子。
- サブジェクトキー識別子。
- 鍵の使用方法。デフォルトでは、デジタル署名 (ビット 0)、否認防止 (ビット 1)、キー証明書サイン (ビット 5)、および CRL サイン (ビット 6) ビットが設定されています。拡張機能は、PKIX 標準および RFC 2459 によって推奨されているとマークされます。Key Usage 拡張機能の説明は、RFC 2459 を参照してください。
- 拡張機能の Base-64 SEQUENCE。これはカスタム拡張用です。MIME 64 DER でエンコードされた形式のエクステンションをテキストフィールドに貼り付けます。

複数の拡張機能を追加するには、**ExtJoiner** プログラムを使用します。ツールの使用方法は、『Certificate System コマンドラインツールガイド』を参照してください。

12. ウィザードはキーペアを生成し、証明書署名要求を表示します。



リクエストは base-64 でエンコードされた PKCS #10 形式であり、マーカーファイル **-----BEGIN NEW CERTIFICATE REQUEST-----** および **-----END NEW CERTIFICATE REQUEST-----** でバインドされます。以下に例を示します。

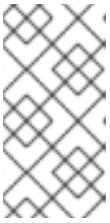
```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICJzCCAZCgAwIBAgIBAzANBgkqhkiG9w0BAQQFADBC6SAwHgYDVQQKExdOZXRzY2F
wZSBDb21tdW5pY2
F0aW9uczngjhnMVQ2VydGlmaWNhdGUgQXV0aG9yaXR5MB4XDTk4MDgyNzE5MDAwMFo
XDTk5MDIyMzE5MDA
wMnbjdgngYoxIDAeBgNVBAoTF05ldHNjYXBlIENvbW11bmljYXRpb25zMQ8wDQYDVQQLEw
ZQZW9wbGUxFz
AVBgoJkiaJkIsZAEBEwdzdXByaXlhMRcwFQYDVQQDEw5TdXByaXlhIFNoZXR0eTEjMCEGC
SqGSIb3Dbndg
JARYUc3Vwcml5Yhvfggsvwryw4y7214vAOBgNVHQ8BAf8EBAMCBLAwFAYJYIZIAYb4QgEB
AQHBAQDAgCAM
A0GCSqGSIb3DQEBBAUAA4GBAFi9FzyJlLmS+kzsue0kTXawbwamGdYql2w4hIBgdR+jWeL
mD4CP4x
-----END NEW CERTIFICATE REQUEST-----
```

ウィザードは、証明書要求を、**/var/lib/pki/*instance_name*/*subsystem_type*/conf/** にある設定ディレクトリーに作成したテキストファイルにコピーします。テキストファイルの名前は、要求された証明書の種類によって異なります。作成される可能性のあるテキストファイルを 表 17.2「証明書署名要求用に作成されたファイル」 に記載します。

表17.2 証明書署名要求用に作成されたファイル

ファイル名	証明書署名要求
kracsr.txt	KRA トランスポート証明書
sslcsr.txt	SSL サーバー証明書
othercsr.txt	Certificate Manager CRL 署名証明書または SSL クライアント証明書などの他の証明書

CA に送信する前に、証明書要求を変更しないでください。リクエストはウィザード経由で自動的に送信することも、クリップボードにコピーして、終了ページを介して CA に手動で送信することもできます。



注記

ウィザードの自動送信機能は、リモートの Certificate Manager にのみ要求を送信できます。サードパーティーの CA に要求を送信するのに使用できません。サードパーティーの CA に要求を送信するには、証明書要求ファイルのいずれかを使用します。

13. 証明書を取得します。

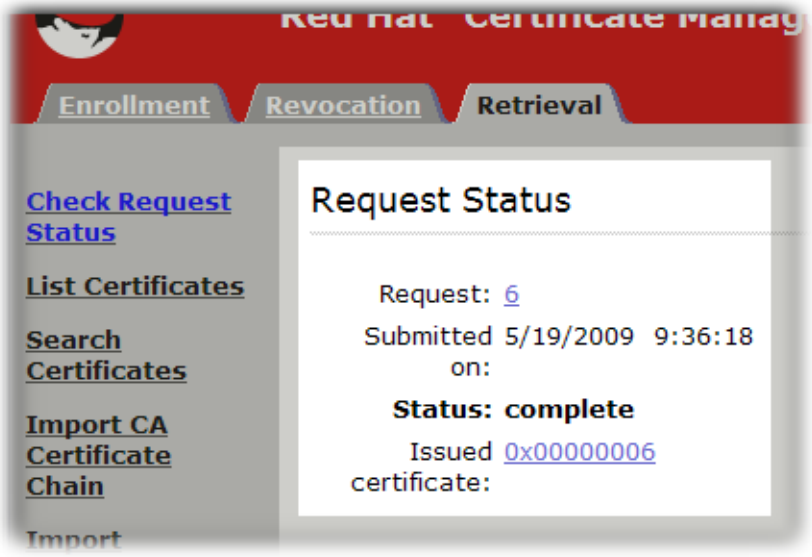
    a. Certificate Manager エンドエンティティーを開きます。

    ```
    https://server.example.com:8443/ca/ee/ca
    ```

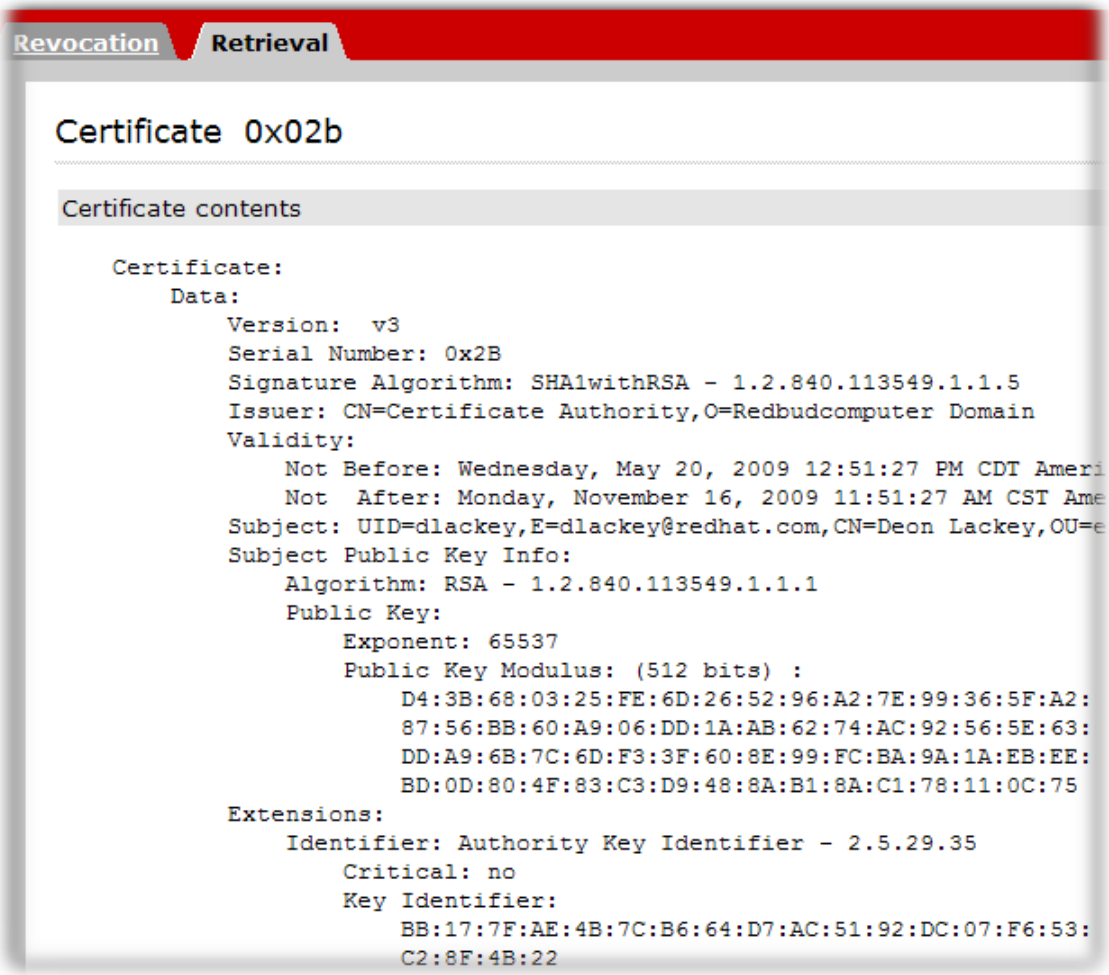
    b. **Retrieval** タブをクリックします。

    c. 証明書要求の送信時に作成された要求 ID 番号を入力し、**Submit** をクリックします。

    d. 次のページには、証明書要求のステータスが表示されます。ステータスが **complete** すると、証明書へのリンクがあります。**Issued certificate** のリンクをクリックします。



e. 新しい証明書情報は、pretty-print 形式、base-64 エンコード形式、および PKCS #7 形式で表示されます。



f. base-64 でエンコードされた証明書 (マーカー行 **-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** を含む) をテキストファイルにコピーします。テキストファイルを保存し、それを使用して証明書のコピーをサブシステムの内部データベースに保存します。「ユーザーの作成」を参照してください。

17.3. サブシステム証明書の更新

証明書を更新する方法は 2 つあります。証明書を **再生成** すると、元の鍵と元のプロファイルと要求を取得し、新しい有効期間と有効期限で同一の鍵を再作成します。証明書の**キーを再入力** すると、最初の証明書要求が元のプロファイルに再送信されますが、新しいキーペアが生成されます。管理者証明書は、キーを再入力することで更新できます。

17.3.1. エンドエンティティーフォームでの証明書のキーの再設定

サブシステム証明書は、元の証明書のシリアル番号を使用して、エンドユーザー登録フォームで直接更新できます。

1. 「証明書の更新」の説明に従って、CA のエンドエンティティー形式で証明書を更新し す。これには、更新するサブシステム証明書のシリアル番号が必要です。

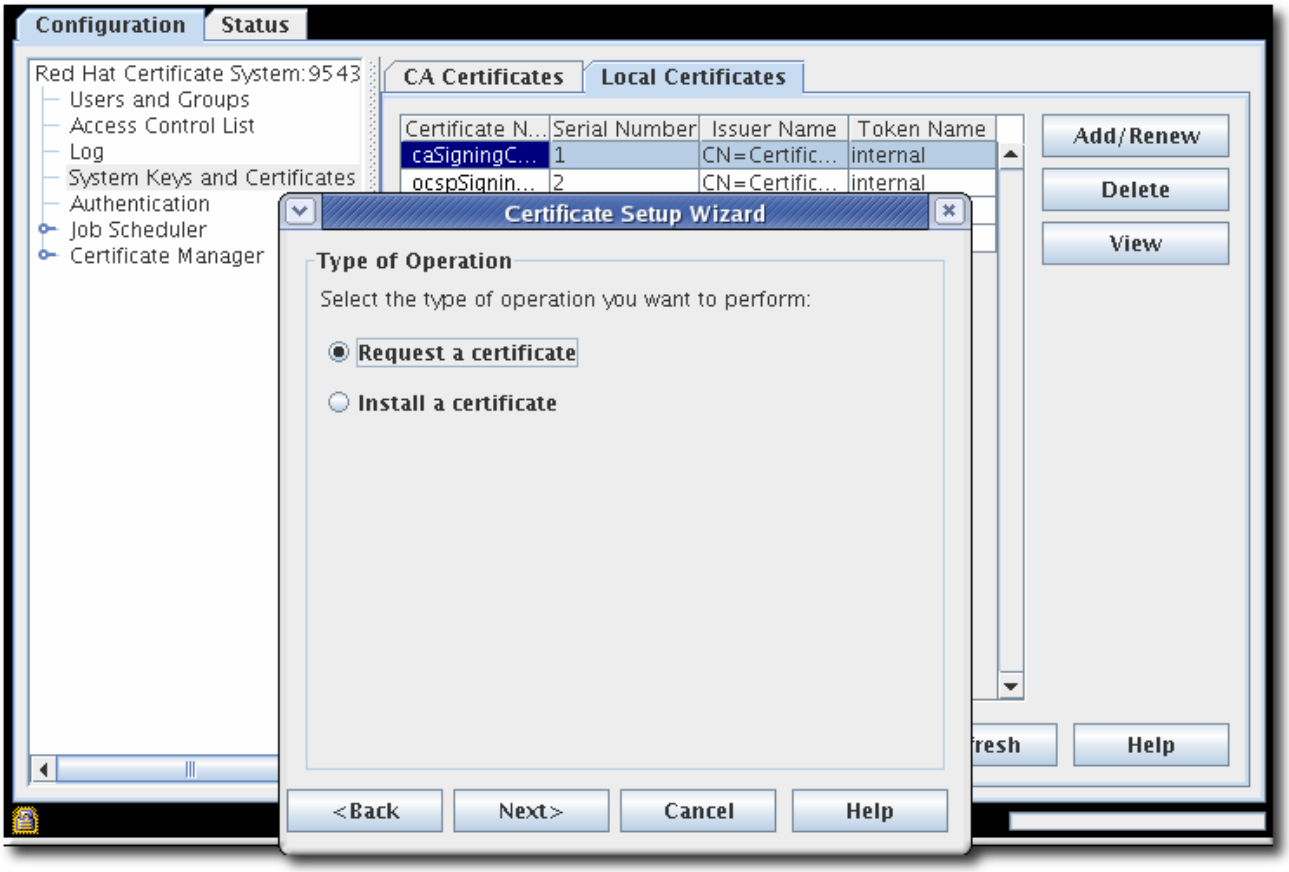
2. 「証明書システムデータベースでの証明書のインストール」の説明に従って、証明書をサブシステムのデータベースにインポートします。証明書は、**certutil** またはコンソールを使用してインポートできます。以下に例を示します。

```
certutil -A -n "ServerCert cert-example" -t u,u,u -d /var/lib/pki/instance_name/alias -a -i
/tmp/example.cert
```

17.3.2. コンソールでの証明書の更新

Java サブシステムは管理コンソールを使用してサブシステム証明書を更新できます。このプロセスは、新しいサブシステム証明書を要求する (「コンソールを使用した証明書の要求」) のと同じですが、重要な違いの 1 つは新しい鍵を生成するのではなく、**既存のキーペア** を使用します。

図17.1 サブシステム証明書の更新



証明書を更新したら、データベース (「データベースからの証明書の削除」) から元の証明書を削除します。

17.3.3. certutil を使用した証明書の更新

certutil は、証明書データベースの既存のキーペアを使用して証明書要求を生成するのに使用できます。その後、新しい証明書要求は、通常のプロファイルページで送信して CA で更新された証明書を発行できます。

注記

暗号化および署名証明書は単一のステップで作成されます。ただし、更新プロセスは一度に 1 つの証明書のみを更新します。

証明書ペアで両方の証明書を更新するには、各証明書を個別に更新する必要があります。

1. トークンデータベースのパスワードを取得します。

```
cat /var/lib/pki/instance_name/conf/password.conf

internal=263163888660
```

2. 証明書を更新しているインスタンスの証明書データベースディレクトリーを開きます。

```
cd /var/lib/pki/instance_name/alias
```

3. 更新する証明書のキーとニックネームを一覧表示します。証明書を更新するには、生成に使用されるキーペアと、新しい証明書に指定されたサブジェクト名は、古い証明書の証明書と同じである必要があります。

```
# certutil -K -d .

certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key and Certificate Services"
Enter Password or Pin for "NSS Certificate DB":
< 0> rsa      69481646e38a6154dc105960aa24ccf61309d37d   caSigningCert cert-pki-tomcat CA
```

4. **alias** ディレクトリーをバックアップとしてコピーし、証明書データベースから元の証明書を削除します。以下に例を示します。

```
certutil -D -n "ServerCert cert-example"  -d .
```

5. 既存の証明書の値にオプションを設定して **certutil** コマンドを実行します。

```
certutil -d . -R -n "NSS Certificate DB:cert-pki-tomcat CA" -s "cn=CA Authority,o=Example Domain" -a -o example.req2.txt
```

新しい証明書とキーのペアの生成と証明書の更新の違いは、**-n** オプションの値です。新しいリクエストとキーのペアを生成するには、**-k** はキータイプを設定してから **-g** で使用します。これは、ビット長を設定します。更新要求では、**-n** オプションは証明書のニックネームを使用してセキュリティーデータベースに保存された既存のキーペアにアクセスします。

パラメーターの詳細は、certutil(1) の man ページを参照してください。

6. 「証明書の要求および受信」 の説明に従って、証明書要求を送信して取得し、インストールします。

17.3.4. システム証明書の更新

Certificate System は、PKI サーバーの実行中にシステム証明書をオンラインに自動的に更新しません。ただし、システム証明書の期限が切れると、証明書システムが起動できません。

システム証明書を更新するには、以下のコマンドを実行します。

1. システム証明書の有効期限が切れている場合は、以下を行います。

   a. 一時証明書を作成します。

```
# pki-server cert-create sslserver --temp
```

b. Certificate System の Network Security Services (NSS) データベースに一時証明書をインポートします。

```
# pki-server cert-import sslserver
```

c. Certificate System を開始します。

```
# pki-server start instance_name
```

2. 証明書を表示し、期限切れのシステム証明書の ID をメモします。

```
# pki-server cert-find
```

3. 新しい永続的な証明書を作成します。

```
# pki-server cert-create certificate_ID
```

4. Certificate System を停止します。

```
# pki-server stop instance_name
```

5. 新しい証明書をインポートして、期限切れの証明書を置き換えます。

```
# pki-server cert-import certificate_ID
```

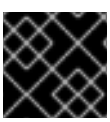
6. Certificate System を開始します。

```
# pki-server start instance_name
```

17.4. サブシステム証明書の名前の変更

証明書の更新方法の 1 つに、新しい証明書に置き換える方法があります。つまり、新しい証明書が新しい鍵で生成されます。通常、新しい証明書をデータベースに追加し、古い証明書を削除して、シンプルな 1 対 1 のスワップに追加できます。これは、個別のサブシステムサーバーがニックネームに基づいて証明書を識別するためです。証明書のニックネームが同じままであれば、サブジェクト名、シリアル番号、キーなどの他の要素が異なる場合でも、サーバーは必要な証明書を見つけることができます。

ただし、状況によっては、新しい証明書に新しい証明書のニックネームが付けられる場合もあります。その場合は、サブシステムの **CS.cfg** 設定ファイルで必要なすべての設定で証明書のニックネームを更新する必要があります。



重要

CS.cfg ファイルの編集後に必ずサブシステムを再起動します。

以下の表では、サブシステムの証明書の各設定パラメーターを一覧表示します。

- 表17.3「CA Certificate Nickname パラメーター」
- 表17.4「KRA Certificate ニックネームパラメーター」

- 表17.5「OCSP Certificate ニックネームパラメーター」
- 表17.6「TKS Certificate ニックネームパラメーター」
- 表17.7「CS.cfg の TPS Nickname パラメーター」

表17.3 CA Certificate Nickname パラメーター

CA 署名証明書	• ca.cert.signing.nickname • ca.signing.cacertnickname • ca.signing.certnickname • ca.signing.nickname • cloning.signing.nickname
OCSP 署名証明書	• ca.ocsp_signing.cacertnickname • ca.ocsp_signing.certnickname • ca.cert.ocsp_signing.nickname • ca.ocsp_signing.nickname • cloning.ocsp_signing.nickname
サブシステム証明書	• ca.cert.subsystem.nickname • ca.subsystem.nickname • cloning.subsystem.nickname • pkiremove.cert.subsystem.nickname
サーバー証明書	• ca.sslserver.nickname • ca.cert.sslserver.nickname
監査署名証明書	• ca.audit_signing.nickname • ca.cert.audit_signing.nickname • cloning.audit_signing.nickname

表17.4 KRA Certificate ニックネームパラメーター

トランスポート証明書	• cloning.transport.nickname • kra.cert.transport.nickname • kra.transport.nickname • tks.kra_transport_cert_nickname このパラメーターは TKS 設定ファイルにあることに注意してください。TKS 証明書がすべて同じままであっても、 **KRA** トランスポート証明書のニックネームが変更された場合は、TKS 設定でこれを変更する必要があります。
ストレージ証明書	• cloning.storage.nickname • kra.storage.nickname • kra.cert.storage.nickname
サーバー証明書	• kra.cert.sslserver.nickname • kra.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • kra.cert.subsystem.nickname • kra.subsystem.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• cloning.audit_signing.nickname • kra.cert.audit_signing.nickname • kra.audit_signing.nickname

表17.5 OCSP Certificate ニックネームパラメーター

OCSP 署名証明書	• cloning.signing.nickname • ocsp.signing.certnickname • ocsp.signing.cacertnickname • ocsp.signing.nickname

サーバー証明書	• ocsp.cert.sslserver.nickname • ocsp.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • ocsp.subsystem.nickname • ocsp.cert.subsystem.nickname • pkiremove.cert.subsystem
監査ログ署名証明書	• cloning.audit_signing.nickname • ocsp.audit_signing.nickname • ocsp.cert.audit_signing.nickname

表17.6 TKS Certificate ニックネームパラメーター

KRA 転送証明書[a]	• tks.kra_transport_cert_nickname
サーバー証明書	• tks.cert.sslserver.nickname • tks.sslserver.nickname
サブシステム証明書	• cloning.subsystem.nickname • tks.cert.subsystem.nickname • tks.subsystem.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• cloning.audit_signing.nickname • tks.audit_signing.nickname • tks.cert.audit_signing.nickname

[a] TKS 証明書がすべて同じままであっても、 **KRA** トランスポート証明書のニックネームが変更された場合は、TKS 設定でこれを変更する必要があります。

表17.7 CS.cfg の TPS Nickname パラメーター

サーバー証明書	• tps.cert.sslserver.nickname
サブシステム証明書	• tps.cert.subsystem.nickname • selftests.plugin.TPSValidity.nickname • selftests.plugin.TPSPresence.nickname • pkiremove.cert.subsystem.nickname
監査ログ署名証明書	• tps.cert.audit_signing.nickname

17.5. ペア間の証明書の使用

1990 年代後半、米国政府が公開鍵インフラストラクチャーを強化し始めたとき、独自の個別の PKI デプロイメントを持つ政府機関は、独自の CA から証明書が発行 **されたかのように**、相互に証明書を認識して信頼できるようにする必要があることが明らかになりました。(外部クライアントが使用するためにネットワークの外部で証明書を信頼する方法は、PKI 管理者にとって深刻で、簡単に解決できない問題です。)

米国政府は、Federal Bridge Certificate Authority と呼ばれる **クロスペア証明書** を発行するための標準を考案しました。これらの証明書は、明白な理由により **ブリッジ証明書** とも呼ばれます。ブリッジ証明書またはクロスペア証明書は、ユーザーの暗号化と署名証明書のペアと同様に、デュアル証明書ペアとしてフレーム化された CA 署名証明書であり、ペア内の各証明書のみが異なる CA によって発行されます。両方のパートナー CA は、その他の CA 署名証明書をデータベースに保存するため、他の PKI 内で発行されたすべての証明書は信頼され、認識されます。

ブリッジング証明書は、独自の PKI でルート CA にチェーンされていない CA によって発行された証明書を尊重します。クロスペア CA 証明書を介して Certificate System CA と他の CA の間に信頼を確立することで、単一の CA 証明書をダウンロードしてインストールすることで CA が発行したすべての証明書を信頼するのと同様に、クロスペア証明書をダウンロードして他の CA が発行した証明書を信頼するために使用することができます。

Certificate System は、クロスペアの CA 証明書を発行、インポート、および公開できます。ペア間の証明書を発行するために特別なプロファイルを作成し、CA サブシステムの Certificate Wizard を使用して CA に対して証明書を要求およびインストールすることができます。

クロスペア証明書プロファイルの作成に関する詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』の『クロスペアプロファイルの設定』セクションを参照してください。

ペア間の証明書の公開に関する詳細は、「ペア間の証明書の公開」 を参照してください。

17.5.1. ペア間の証明書のインストール

両方のクロスペア証明書は、「証明書システムデータベースでの証明書のインストール」 に記載されているように、**certutil** ツールを使用して、または Certificate Setup Wizard から **Cross-Pair Certificates** オプションを選択して、Certificate System データベースにインポートすることができます。

両方の証明書がデータベースにインポートされると、**crossCertificatePair** エントリーが形成され、データベースに保存されます。**crossCertificatePair** エントリーが作成されると、元の個々のクロスペアの CA 証明書が削除されます。

17.5.2. ペア間の証明書の検索

ブリッジ証明書内の両方の CA は、クロスのペア証明書を LDAP データベースの **crossCertificatePair** エントリーとして保存または公開することができます。**ldapsearch** を使用して、Certificate Manager の内部データベースで **crossCertificatePair** エントリーを検索できます。

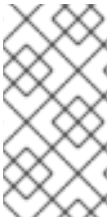
```
/usr/lib[64]/mozldap/ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -
b "o=server.example.com-pki-ca" -s sub "(crossCertificatePair=*)"
```

17.6. 証明書データベースの管理

各 Certificate System インスタンスには、内部トークンで維持される証明書データベースがあります。このデータベースには、Certificate System インスタンスにインストールされているサブシステムに属する証明書と、サブシステムが受信する証明書の検証に使用する各種 CA 証明書が含まれます。

外部トークンを使用してキーペアを生成および保存する場合でも、Certificate System は常に、信頼できる CA 証明書と信頼できない CA 証明書のリストを内部トークンに保持します。

本セクションでは、Certificate System ウィンドウを使用して、証明書データベースの内容を表示する方法、不要な証明書を削除する方法、およびデータベースにインストールされている CA 証明書の信頼設定を変更する方法を説明します。データベースに証明書を追加する方法は、「証明書システムデータベースでの証明書のインストール」 を参照してください。



注記

Certificate System コマンドラインユーティリティー **certutil** は、信頼設定を編集し、証明書を追加または削除して、証明書データベースを管理するのに使用できます。このツールの詳細は、http://www.mozilla.org/projects/security/pki/nss/tools/ を参照してください。

管理者は、証明書データベースの内容を定期的にチェックして、不要な CA 証明書が含まれていないことを確認する必要があります。たとえば、データベースに PKI セットアップ内で信頼されるべきではない CA 証明書が含まれている場合は、それらを削除します。

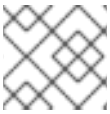
17.6.1. 証明書システムデータベースでの証明書のインストール

サブシステムに対して新しいサーバー証明書が発行された場合は、そのサブシステムデータベースにインストールする必要があります。さらに、ユーザー証明書およびエージェント証明書をサブシステムデータベースにインストールする必要があります。証明書が外部 CA により発行される場合は、通常、対応する CA 証明書または証明書チェーンをインストールする必要があります。

証明書は、Console の Certificate Setup Wizard を使用するか、または **certutil** ユーティリティーを使用してサブシステム証明書データベースにインストールできます。

- 「コンソールを使用した証明書のインストール」
- 「certutil を使用した証明書のインストール」
- 「CA 証明書のチェーンについて」

17.6.1.1. コンソールを使用した証明書のインストール



注記

pkiconsole が非推奨になりました。

Certificate Setup Wizard は、Certificate System インスタンスが使用する内部トークンまたは外部トークンのいずれかに以下の証明書をインストールまたはインポートできます。

- Certificate System サブシステムが使用する証明書のいずれか
- 外部 CA またはその他の Certificate System CA からの信頼済み CA 証明書
- 証明書チェーン

証明書チェーンには、証明書のコレクション (サブジェクト証明書、信頼されたルート CA 証明書、およびサブジェクト証明書を信頼されたルートにリンクするために必要な中間 CA 証明書) が含まれます。ただし、ウィザードがインポートする証明書チェーンには、CA 証明書のみを含める必要があります。どの証明書もユーザー証明書にすることはできません。

証明書チェーンでは、チェーンの各証明書は個別の DER でエンコードされたオブジェクトとしてエンコードされます。ウィザードが証明書チェーンをインポートすると、これらのオブジェクトが次々にインポートされ、チェーンの最後の証明書 (ルート CA 証明書である場合とそうでない場合があります) までインポートされます。チェーン内の証明書のいずれかがローカル証明書データベースにすでにインストールされている場合、ウィザードは既存の証明書をチェーン内の証明書に置き換えます。チェーンに中間 CA 証明書が含まれる場合、ウィザードは *信頼されていない* CA 証明書として証明書データベースに追加します。

サブシステムコンソールは、同じウィザードを使用して証明書と証明書チェーンをインストールします。ローカルセキュリティーデータベースに証明書をインストールするには、以下の手順を実施します。

1. コンソールを開きます。

   ```
   pkiconsole https://server.example.com:secure_port/subsystem_type
   ```

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. サブシステムのタイプと証明書のタイプに応じて、証明書をインストールできる 2 つのタブがあります。

   - **CA 証明書** タブは、CA 証明書および証明書チェーンのインストールに使用されます。Certificate Manager の場合、このタブはサードパーティーの CA 証明書またはその他の Certificate System CA 証明書に使用されます。すべてのローカル CA 証明書は、**Local Certificates** タブにインストールされます。その他のサブシステムでは、すべての CA 証明書およびチェーンがこのタブでインストールされます。
   - **Local Certificates** タブには、すべてのサーバー証明書、サブシステム証明書、および OCSP 署名や KRA トランスポートなどのローカル証明書がインストールされます。

   該当するタブを選択します。
4. **Local Certificates** タブで証明書をインストールするには、**Add/Renew** をクリックします。**CA Certificates** タブに証明書をインストールするには、**Add** をクリックします。いずれも Certificate Setup Wizard を開きます。

a. ウィザードが開いたら、**Install a certificate** ラジオボタンを選択し、**Next** をクリックします。

b. インストールする証明書のタイプを選択します。ドロップダウンメニューのオプションは、サブシステムのタイプに応じて、証明書の作成に使用できるオプションと同じですが、クロスペア証明書をインストールするための追加オプションがあります。

c. **-----BEGIN CERTIFICATE-----** および **-----END CERTIFICATE-----** を含む証明書の本文にテキストエリアに貼り付けるか、絶対ファイルの場所を指定します。これはローカルファイルでなければなりません。

証明書は以下のようになります。

```
-----BEGIN CERTIFICATE-----
MIICKzCCAZSgAwIBAgIBAzANgkqkiG9w0BAQQFADA3MQswCQYDVQQGEw
JVUzERMA8GA1UEChMITmV0c2NhcGUxFTATBgNVBAsTDFN1cHJpeWEncy
BDQTAeFw05NzEwMTgwMTM2MjVaFw05OTEwMTgwMTM2MjVaMEgxCzAJBg
NVBAYTAlVTMREwDwYDVQQKEwhOZXRzY2FwZTENMAsGA1UECxMEUHawcz
EXMBUGA1UEAxMOU3VwcmI5YSBTaGV0dHkwgZ8wDQYJKoZIhdfNAQEBBQ
ADgY0AMIGJAoGBAMr6eZiPGfjX3uRJgEjmKiqG7SdATYzBcABu1AVyd7
chRFOGD3wNktbf6hRo6EAmM5R1Askzf8AW7LiQZBcrXpc0k4du+2j6xJ
u2MPm8WKuMOTuvzpo+SGXelmHVChEqooCwfdiZywyZNmgaMa2MS6pUkf
QVAgMBAAGjNjA0MBEGCWCGSAGG+EIBAQQEAwIAgD
-----END CERTIFICATE-----
```

5. ウィザードに、証明書の詳細が表示されます。指紋を確認して、これが正しい証明書であることを確認するか、**Back** ボタンをクリックして戻って別のボタンを送信します。証明書のニックネームを指定します。

このウィザードは、証明書をインストールします。

6. 証明書に署名した CA はサブシステムによって信頼される必要があります。この CA の証明書がサブシステムの証明書データベース (内部または外部) に存在し、信頼されていることを確認してください。

CA 証明書がリストされていない場合は、信頼できる CA として証明書を証明書データベースに追加します。CA の証明書がリストされているが信頼されていない場合は、「CA 証明書の信頼設定の変更」 に示すように、信頼設定を信頼済みに変更します。

Certificate System 証明書データベースに保存されていない CA によって発行された証明書をインストールする場合は、その CA の証明書チェーンをデータベースに追加します。CA チェーンをデータベースに追加するには、CA チェーンをテキストファイルにコピーし、ウィザードを再起動して、CA チェーンをインストールします。

17.6.1.2. certutil を使用した証明書のインストール

certutil を使用して Certificate System インスタンスのセキュリティーデータベースにサブシステム証明書をインストールするには、以下を行います。

1. サブシステムのセキュリティーデータベースディレクトリーを開きます。

```
cd /var/lib/pki/instance_name/alias
```

2. **-A** を指定して **certutil** コマンドを実行して、証明書と、CA が発行した証明書が含まれるファイルを示す **-i** を追加しします。

```
certutil -A -n cert-name -t trustargs
  -d . -a -i certificate_file
```

注記

Certificate System インスタンスの証明書およびキーが HSM に保存されている場合は、**-h** オプションを使用してトークン名を指定します。

以下に例を示します。

```
certutil -A -n "ServerCert cert-instance_name" -t u,u,u -d . -a -i /tmp/example.cert
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

17.6.1.3. CA 証明書のチェーンについて

証明書をサポートするクライアントまたはサーバーソフトウェアは、証明書データベースに信頼できる CA 証明書のコレクションを保持します。これらの CA 証明書は、ソフトウェアが検証できるその他の証明書を決定します。最も単純なケースでは、ソフトウェアは、証明書を持っている CA の 1 つによって発行された証明書のみを検証できます。信頼できる CA 証明書を CA 証明書のチェーンの一部にすることもできます。各証明書は、証明書階層の上位にある CA によって発行されます。

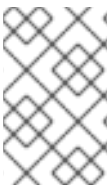
チェーンの最初の証明書は、コンテキスト固有の方法で処理されます。コンテキスト固有の方法は、インポート方法によって異なります。Mozilla Firefox の場合、この処理は、ダウンロードされるオブジェクトで使用される MIME コンテンツタイプによって異なります。Red Hat サーバーでは、サーバー管理インターフェイスで選択したオプションによって異なります。

後続の証明書はすべて同じ扱われます。証明書の Netscape Certificate Type 証明書拡張に SSL-CA ビットが含まれていて、ローカル証明書データベースにまだ存在しない場合、それらは信頼されていない CA として追加されます。チェーンのどこかに信頼できる CA が存在する限り、証明書チェーンの検証に使用できます。

17.6.2. データベースコンテンツの表示

サブシステム証明書データベースに格納されている証明書 **cert9.db** は、サブシステム管理コンソールから表示できます。または、**certutil** ユーティリティーを使用して一覧表示できます。**certutil** は、TPS サブシステムは管理コンソールを使用しないため、TPS 証明書を表示するのに使用する必要があります。

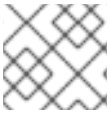
- 「コンソールを使用したデータベースコンテンツの表示」
- 「certutil を使用したデータベースコンテンツの表示」



注記

cert9.db に一覧表示されている証明書データベースは、サブシステム操作に使用されるサブシステム証明書です。ユーザー証明書は、LDAP 内部データベースのユーザーエントリーと共に保存されます。

17.6.2.1. コンソールを使用したデータベースコンテンツの表示



注記

pkiconsole が非推奨になりました。

管理コンソールを使用してデータベースの内容を表示するには、以下を行います。

1. サブシステムコンソールを開きます。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

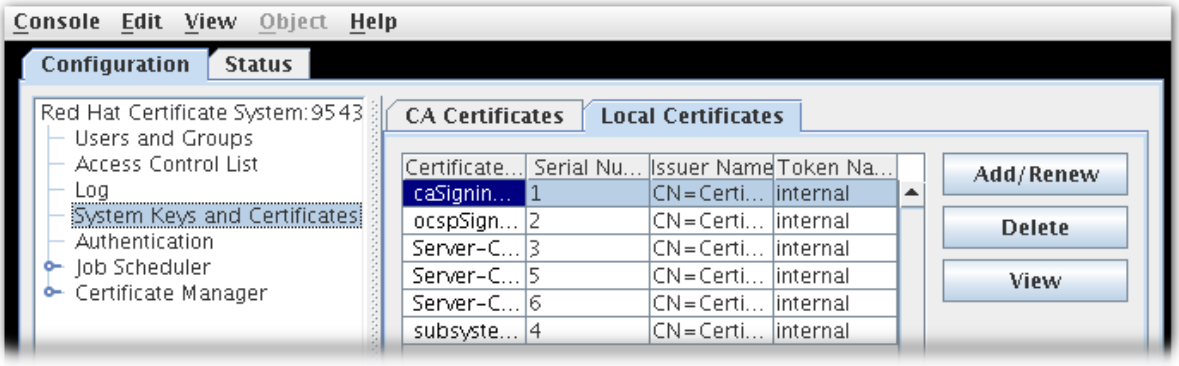
2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。

3. **CA Certificates** および **Local Certificates** の 2 つのタブがあります。ここには、さまざまな種類の証明書が一覧表示されます。

   - **CA 証明書** には、Entrust や Verisign などのサードパーティー CA によって発行された証明書や、外部の Certificate System Certificate Manager など、対応する秘密鍵の資料が利用できない CA 証明書が一覧表示されます。

   - **ローカル証明書** には、KRA トランスポート証明書や OCSP 署名証明書など、Certificate System サブシステムインスタンスによって保持されている証明書が一覧表示されます。

   図17.2 Certificate Database タブ



4. **Certificate Database Management** テーブルには、サブシステムにインストールされている証明書が一覧表示されます。証明書ごとに、以下の情報が提供されます。

   - **Certificate Name**

   - **Serial Number**

   - **Issuer Names**。この証明書の発行者の共通名 (**cn**) です。

   - **Token Name** (証明書を保持する暗号化トークンの名前)。データベースに保存されている証明書の場合、これは **internal** になります。

証明書に関する詳細情報を表示するには、証明書を選択して **View** をクリックします。これにより、証明書のシリアル番号、有効期間、サブジェクト名、発行者名、および証明書のフィンガープリントを表示するウィンドウが開きます。

17.6.2.2. certutil を使用したデータベースコンテンツの表示

certutil を使用してサブシステムデータベース内の証明書を表示するには、インスタンスの証明書データベースディレクトリーを開き、**-L** オプションを付けて **certutil** をインストールします。以下に例を示します。

```
cd /var/lib/pki/instance_name/alias

certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

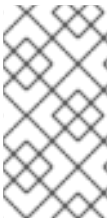
certutil を使用してサブシステムデータベースに保存されている鍵を表示するには、**-K** オプションを指定して **certutil** を実行します。以下に例を示します。

```
cd /var/lib/pki/instance_name/alias

certutil -K -d .

Enter Password or Pin for "NSS Certificate DB":
<0> subsystemCert cert-instance_name
<1>
<2> Server-Cert cert-instance_name
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

17.6.3. データベースからの証明書の削除

不要な証明書を削除すると、証明書データベースのサイズが減少します。



注記

証明書データベースから CA 証明書を削除するときは、*中間 CA 証明書* を削除しないように注意してください。これにより、サブシステムが信頼できる CA 証明書にチェーンアップするのに役立ちます。不明な場合は、データベースの証明書を *信頼されていない* CA 証明書として残します。「CA 証明書の信頼設定の変更」 を参照してください。

- 「コンソールを使用した証明書の削除」
- 「certutil を使用した証明書の削除」

17.6.3.1. コンソールを使用した証明書の削除

注記

pkiconsole が非推奨になりました。

コンソールから証明書を削除するには、以下の手順を実施します。

1. サブシステムコンソールを開きます。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. 削除する証明書を選択して、**Delete** をクリックします。
4. プロンプトが表示されたら、削除を確定します。

17.6.3.2. certutil を使用した証明書の削除

certutil を使用してデータベースから証明書を削除するには、以下を実行します。

1. インスタンスの証明書データベースディレクトリーを開きます。

```
/var/lib/pki/instance_name/alias
```

2. **-L** オプションを使用して **certutil** を実行し、データベースの証明書を一覧表示します。以下に例を示します。

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

3. **-D** オプションを指定して **certutil** を実行し、証明書を削除します。

```
certutil -D -d . -n certificate_nickname
```

以下に例を示します。

```
certutil -D -d . -n "ServerCert cert-instance_name"
```

4. 証明書を再度一覧表示し、証明書が削除されたことを確認します。

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

17.7. CA 証明書の信頼設定の変更

Certificate System サブシステムは、証明書データベース内の CA 証明書を使用して、SSL 対応の通信中に受信した証明書を検証します。

証明書データベースに保存されている CA の信頼設定を、一時的または永続的に変更する必要がある場合があります。たとえば、アクセスまたは侵害された証明書に問題がある場合、CA 証明書を信頼できないものとしてマークすると、その CA によって署名された証明書を持つエンティティーが Certificate System に対して認証されなくなります。問題が解決されると、CA は再び信頼できるとマークできます。

CA の信頼を永続的に解除するには、信頼データベースからその証明書を削除することを検討してください。手順は、「データベースからの証明書の削除」を参照してください。

17.7.1. コンソールからの信頼設定の変更

注記

pkiconsole が非推奨になりました。

CA 証明書の信頼設定を変更するには、以下を実行します。

1. サブシステムコンソールを開きます。

```
pkiconsole https://server.example.com:secure_port/subsystem_type
```

2. **Configuration** タブで、左側のナビゲーションツリーから **System Keys and Certificates** を選択します。
3. **CA certificates** タブを選択します。
4. 変更する CA 証明書を選択し、**Edit** をクリックします。
5. **The Certificate chain is (un)trusted, are you sure you want to (un)trust it?** というプロンプトが開きます。

   yes をクリックすると、証明書チェーンの信頼設定が変更されます。**no** を押すと、元の信頼関係が保持されます。

17.7.2. certutil を使用した信頼設定の変更

certutil を使用して証明書の信頼設定を変更するには、以下を実行します。

1. インスタンスの証明書データベースディレクトリーを開きます。

```
cd /var/lib/pki/instance_name/alias
```

2. **-L** オプションを使用して **certutil** を実行し、データベースの証明書を一覧表示します。以下に例を示します。

```
certutil -L -d .

Certificate Authority - Example Domain    CT,c,
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

3. **-M** オプションを使用して **certutil** を実行し、証明書の信頼設定を変更します。

```
certutil -M -n cert_nickname -t trust -d .
```

   以下に例を示します。

```
certutil -M -n "Certificate Authority - Example Domain" -t TCu,TCu,TCu -d .
```

4. 証明書を再度一覧表示し、証明書が削除されたことを確認します。

```
certutil -L -d .

Certificate Authority - Example Domain    CTu,CTu,CTu
subsystemCert cert-instance_name          u,u,u
Server-Cert cert-instance_name            u,u,u
```

certuil コマンドの使用方法は、http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html を参照してください。

17.8. サブシステムによって使用されるトークンの管理

Certificate System は、トークンの 2 つのグループを管理します。PKI タスクを実行するためにサブシステムによって使用されるトークンと、サブシステムを通じて発行されるトークンです。これらの管理タスクは、特にサブシステムによって使用されるトークンを参照します。

スマートカードトークンの管理方法は、*6章Token Management System の使用および設定: TPS および TKS* を参照してください。

17.8.1. トークンの検出

インストールまたは設定する Certificate System によってトークンを検出できるかどうかを確認するには、**TokenInfo** ユーティリティーを使用します。

```
TokenInfo /var/lib/pki/instance_name/alias
Database Path: /var/lib/pki/instance_name/alias
Found external module 'NSS Internal PKCS #11 Module'
```

このユーティリティーは、Certificate System にインストールされたトークンだけでなく、Certificate System で検出できるすべてのトークンを返します。

17.8.2. トークンの表示

Certificate System インスタンスに現在インストールされているトークンの一覧を表示するには、**modutil** ユーティリティーを使用します。

1. インスタンスの **alias** ディレクトリーを開きます。以下に例を示します。

```
cd /var/lib/pki/instance_name/alias
```

2. インストールされている PKCS #11 モジュールに関する情報と、**modutil** ツールを使用して、対応するトークンに関する情報を表示します。

```
modutil -dbdir . -nocertdb -list
```

17.8.3. トークンのパスワードの変更

サブシステムのキーペアと証明書を格納する内部または外部のトークンは、パスワードによって保護 (暗号化) されます。キーペアを復号する、またはキーペアにアクセスするには、トークンのパスワードを入力します。このパスワードは、トークンが最初にアクセスされたとき、通常は Certificate System のインストール時に設定されます。

サーバーのキーと証明書を保護するパスワードを定期的に変更することをお勧めします。パスワードを変更すると、誰かがパスワードを見つけるリスクを最小限に抑えることができます。トークンのパスワードを変更するには、**certutil** コマンドラインユーティリティーを使用します。

certutil の詳細は、http://www.mozilla.org/projects/security/pki/nss/tools/ を参照してください。

シングルサインオンパスワードキャッシュは、**password.conf** ファイルにトークンパスワードを保存します。このファイルは、トークンパスワードが変更されるたびに手動で更新する必要があります。**password.conf** ファイルを介したパスワード管理の詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 を参照してください。

第18章 RED HAT ENTERPRISE LINUX 7 での日時の設定

このセクションは、Red Hat Enterprise Linux 7 で日時を設定する方法を説明します。

システム時間は常に **協定世界時** (UTC) で維持され、必要に応じてアプリケーション内でローカル時間に変換されます。**ローカルタイム** は、**夏時間** (DST) を考慮に入れた現行タイムゾーンの実際の時刻です。

timedatectl ユーティリティーは、**systemd** システムおよびサービスマネージャーの一部として配布されており、システムクロック設定を確認および変更できます。

システムの現在時刻の変更

```
timedatectl set-time HH:MM:SS
```

HH は時間、*MM* は分、*SS* は秒 (すべて 2 桁) の数字に置き換えます。

現在日の変更

```
timedatectl set-time YYYY-MM-DD
```

YYYY は 4 桁の年に、*MM* と *DD* は 2 桁の月と日に置き換えます。

この時間の変更はオペレーティングシステムによって監査されます。詳細は、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『時間変更イベントの監査』 セクションを参照してください。

第19章 証明書システムの製品バージョンの特定

Red Hat Certificate System の製品バージョンは、**/usr/share/pki/CS_SERVER_VERSION** ファイルに保存されます。バージョンを表示するには、以下を実行します。

```
# cat /usr/share/pki/CS_SERVER_VERSION
Red Hat Certificate System 10.0 (Batch Update 1)
```

実行中のサーバーの製品バージョンを検索するには、ブラウザーから以下の URL にアクセスします。

- **http://*host_name*:*port_number*/ca/admin/ca/getStatus**
- **http://*host_name*:*port_number*/kra/admin/kra/getStatus**
- **http://*host_name*:*port_number*/ocsp/admin/ocsp/getStatus**
- **http://*host_name*:*port_number*/tks/admin/tks/getStatus**
- **http://*host_name*:*port_number*/tps/admin/tps/getStatus**

注記

各コンポーネントは個別のパッケージであるため、バージョン番号は異なります。上記は、現在実行中の各コンポーネントのバージョン番号を示しています。

第20章 RED HAT CERTIFICATE SYSTEM の更新

Certificate System と、それが稼働しているオペレーティングシステムを更新するには、**yum update** コマンドを使用します。これにより、Certificate System とオペレーティングシステムパッケージの更新がダウンロード、検証、およびインストールされます。証明書システムの更新および更新が成功したことの検証の詳細については、『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『Certificate System パッケージの更新』 セクションを参照してください。

第21章 トラブルシューティング

この章では、Certificate System のインストール時に発生する一般的な使用上の問題のいくつかを説明します。

問： init スクリプトは OK ステータスを返しましたが、その CA インスタンスは応答しません。理由

答： これは起こらないはずです。通常 (常にではありませんが)、これは CA のリスナーの問題を示しますが、さまざまな原因が考えられます。**catalina.out**、**system**、**およびデバッグ** ログファイルでインスタンスに対して、発生したエラーを確認します。これは、いくつかの共通エラーを示しています。

1 つの状況は、CA の PID があり、プロセスが実行されているが、サーバーのリスナーが開かれていないことを示している場合です。これにより、Java 呼び出しクラスエラーが **catalina.out** ファイルに返されます。

```
Oct 29, 2010 4:15:44 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-9080
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:64)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:615)
        at org.apache.catalina.startup.Bootstrap.load(Bootstrap.java:243)
        at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:408)
Caused by: java.lang.UnsatisfiedLinkError: jss4
```

これは、JSS または NSS の誤ったバージョンがあることを意味します。プロセスに **libnss3.so** が必要です。以下のコマンドでこれを確認します。

```
ldd /usr/lib64/libjss4.so
```

libnss3.so が見つからない場合は、**LD_LIBRARY_PATH** 変数の設定を解除し、CA を再起動します。

```
unset LD_LIBRARY_PATH
pki-server restart instance_name
```

問： pkiconsole を開くことができません。標準出力 (stdout) で Java の例外が見られます。

答： これはおそらく、間違った JRE がインストールされているか、間違った JRE がデフォルトとして設定されていることを意味します。**alternatives --config java** を実行して、選択した JRE を確認します。Red Hat Certificate System には OpenJDK 1.8 が必要です。

問： pkiconsole の実行を試みましたが、標準出力 (stdout) でソケット例外が取得されました。理由

答： これは、ポートに問題があることを意味します。管理ポートの SSL 設定が間違っている (**server.xml** の設定が間違っている) か、管理者インターフェイスにアクセスするために間違ったポートが付与されたかのいずれかです。

ポートエラーは以下のようになります。

```
NSS Cipher Supported '0xff04'
java.io.IOException: SocketException cannot read on socket
        at org.mozilla.jss.ssl.SSLSocket.read(SSLSocket.java:1006)
        at org.mozilla.jss.ssl.SSLInputStream.read(SSLInputStream.java:70)
        at
com.netscape.admin.certsrv.misc.HttpInputStream.fill(HttpInputStream.java:303)
        at
com.netscape.admin.certsrv.misc.HttpInputStream.readLine(HttpInputStream.java:224)
        at
com.netscape.admin.certsrv.connection.JSSConnection.readHeader(JSSConnection.java:439)
        at
com.netscape.admin.certsrv.connection.JSSConnection.initReadResponse(JSSConnection.java:4
30)
        at
com.netscape.admin.certsrv.connection.JSSConnection.sendRequest(JSSConnection.java:344)

        at
com.netscape.admin.certsrv.connection.AdminConnection.processRequest(AdminConnection.java
:714)
        at
com.netscape.admin.certsrv.connection.AdminConnection.sendRequest(AdminConnection.java:62
3)
        at
com.netscape.admin.certsrv.connection.AdminConnection.sendRequest(AdminConnection.java:59
0)
        at
com.netscape.admin.certsrv.connection.AdminConnection.authType(AdminConnection.java:323)

        at
com.netscape.admin.certsrv.CMSServerInfo.getAuthType(CMSServerInfo.java:113)
        at com.netscape.admin.certsrv.CMSAdmin.run(CMSAdmin.java:499)
        at com.netscape.admin.certsrv.CMSAdmin.run(CMSAdmin.java:548)
        at com.netscape.admin.certsrv.Console.main(Console.java:1655)
```

問： 証明書を登録しようとしたら request is not submitted...Subject Name Not Found というエラーが表示される

答： これは、カスタム LDAP ディレクトリー認証プロファイルで最も頻繁に発生し、ディレクトリー操作が失敗したことを示しています。特に、作業 DN を作成できないために失敗しました。このエラーは CA の **debug** ログに表示されます。たとえば、このプロファイルは、ディレクトリーを認識しないカスタム属性 (***MYATTRIBUTE***) を使用します。

```
[14/Feb/2011:15:52:25][http-1244-Processor24]: BasicProfile: populate() policy
setid =userCertSet
[14/Feb/2011:15:52:25][http-1244-Processor24]: AuthTokenSubjectNameDefault:
populate start
[14/Feb/2011:15:52:25][http-1244-Processor24]: AuthTokenSubjectNameDefault:
java.io.IOException: Unknown AVA keyword 'MYATTRIBUTE'.
[14/Feb/2011:15:52:25][http-1244-Processor24]: ProfileSubmitServlet: populate
Subject Name Not Found
[14/Feb/2011:15:52:25][http-1244-Processor24]: CMSServlet: curDate=Mon Feb 14
15:52:25 PST 2011 id=caProfileSubmit time=13
```

サブジェクト DN で使用されるカスタムコンポーネント (属性、オブジェクトクラス、および未登録の OID) は、障害を引き起こす可能性があります。ほとんどの場合、RHC 2253 で定義されている X.509 属性は、カスタム属性ではなくサブジェクト DN で使用する必要があります。

問： 登録した証明書が公開されないのはなぜですか。

答： これは通常 CA の設定が間違っていることを示しています。エラーを検索する主要な場所は **debug** ログで、設定が間違っている場所を示しています。たとえば、以下にはマッパーに関連する問題があります。

```
[31/Jul/2010:11:18:29][Thread-29]: LdapSimpleMap: cert subject
dn:UID=me,E=me@example.com,CN=yes
[31/Jul/2010:11:18:29][Thread-29]: Error mapping:
mapper=com.netscape.cms.publish.mappers.LdapSimpleMap@258fdcd0 error=Cannot
find a match in the LDAP server for certificate. netscape.ldap.LDAPException:
error result (32); matchedDN = ou=people,c=test; No such object
```

CA の **CS.cfg** ファイル、または CA コンソールの タブで **Publishing** 設定を確認します。この例では、この問題はマッピングパラメーターにあり、**既存の** LDAP 接尾辞を指している必要があります。

```
ca.publish.mapper.instance.LdapUserCertMap.dnPattern=UID=$subj.UID,dc=publish
```

問： リモートホストから pkiconsole ユーティリティーを開くにはどうすればいいですか

答： 特定の状況では、管理者が、リモートホストからの Certificate System サーバーに **pkiconsole** を開く場合があります。このため、管理者は Virtual Network Computing (VNC) 接続を使用できます。

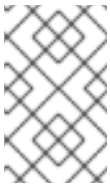
1. Red Hat Certificate System サーバーなどで VNC サーバーを設定します。リモートデスクトップアクセスの詳細については、RHEL 8 ドキュメントの 関連セクション を参照してください。

重要

pkiconsole ユーティリティーは、Federal Information Processing Standard (FIPS) モードが有効になっているサーバーでは実行できません。Certificate System サーバーで FIPS モードが有効になっている場合は、Red Hat Enterprise Linux で別のホストを使用して VNC サーバーを実行します。このユーティリティーは非推奨になることに注意してください。

2. VNC ウィンドウで **pkiconsole** ユーティリティーを開きます。以下に例を示します。

```
# pkiconsole https://server.example.com:8443/ca
```



注記

VNC ビューアーは、さまざまなオペレーティングシステムで使用できます。ただし、Red Hat は、統合リポジトリーから Red Hat Enterprise Linux にインストールされた VNC ビューアーのみをサポートします。

問： LDAP サーバーが応答しないときにどうすればよいですか。

答： 内部データベースに使用されている RedHat Directory Server インスタンスが実行していない場合、接続の問題が発生した場合、または TLS 接続障害が発生した場合、それに依存するサブシステムインスタンスに接続できません。インスタンスのデバッグログは、特に LDAP 接続の問題を特定します。たとえば、LDAP サーバーがオンラインではない場合は、以下のようになります。

```
[02/Apr/2019:15:55:41][authorityMonitor]: authorityMonitor: failed to get LDAPConnection.
Retrying in 1 second.
[02/Apr/2019:15:55:42][authorityMonitor]: In LdapBoundConnFactory::getConn()
[02/Apr/2019:15:55:42][authorityMonitor]: masterConn is null.
[02/Apr/2019:15:55:42][authorityMonitor]: makeConnection: errorIfDown true
[02/Apr/2019:15:55:42][authorityMonitor]: TCP Keep-Alive: true
java.net.ConnectException: Connection refused (Connection refused)
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
    at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
[02/Apr/2019:15:55:42][authorityMonitor]: Can't create master connection in
LdapBoundConnFactory::getConn!
    Could not connect to LDAP server host example911.redhat.com port 389 Error
netscape.ldap.LDAPException:
        Unable to create socket: java.net.ConnectException: Connection refused (Connection
refused) (-1)
```

ケーブルが抜かれた、Red Hat Directory Server が停止した、重大なパケット損失が発生した、または TLS 接続を再作成できることを確認したなど、根本的なネットワークの問題を修正した後、問題の Certificate System インスタンスを停止して開始します。

```
# systemctl stop pki-tomcatd-nuxwdog@instance_name.service
```

```
# systemctl start pki-tomcatd-nuxwdog@instance_name.service
```

第22章 サブシステムの制御およびメンテナンス

この章では、Red Hat Certificate System サブシステムを制御 (開始、停止、再起動、およびステータスチェック) する方法と、一般的なメンテナンス (ヘルスチェック) の推奨事項を説明します。

22.1. 起動、停止、再起動、およびステータスの取得

Red Hat Certificate System サブシステムインスタンスは、Red Hat Enterprise Linux 8 の **systemctl** ユーティリティーを使用して停止および開始できます。

注記

また、**pki-server** エイリアスを使用してインスタンスを開始および停止することもできます。**pki-server <command> <instance>** は、**systemctl <command> pki-tomcatd@<instance>.service.** のエイリアスです。

インスタンスを起動するには、以下のコマンドを実行します。

```
# systemctl start unit_file@instance_name.service
```

```
# pki-server start instance_name
```

インスタンスを停止するには、以下のコマンドを実行します。

```
# systemctl stop unit_file@instance_name.service
```

```
# pki-server stop instance_name
```

インスタンスを再起動するには、以下のコマンドを実行します。

```
# systemctl restart unit_file@instance_name.service
```

```
# pki-server restart instance_name
```

インスタンスのステータスを表示するには、以下のコマンドを実行します。

```
# systemctl status unit_file@instance_name.service
```

unit_file には、以下のいずれかの値を使用できます。

- **pki-tomcat**: ウォッチドッグが無効になっている場合
- **pki-tomcat-nuxwdog**: ウォッチドッグが有効な場合

22.2. サブシステムのヘルスチェック

管理者は、次のような考えられる障害を定期的に監視することが重要です。

- 完全なディスクが起因する監査障害
- HSM 接続の問題が原因での署名失敗

- LDAP サーバー接続の問題
- などになります。

セルフテストは、「セルフテストの実行」で説明されているように要求で実行することもできます。

22.2.1. PKI の Healthcheck

PKI Healthcheck は、Certificate System 環境の状態に影響を与える可能性のある問題を見つけるのに役立つコマンドラインツールです。必要に応じて、このツールは、Red Hat Identity Management に存在する Healthcheck ツールに報告できます。

22.2.1.1. PKI Healthcheck テストモジュール

PKI Healthcheck は、以下をテストする独立したモジュールで設定されています。

- **CS.cfg と NSS データベースとの間の証明書同期**

  (**/var/lib/pki/<instance>/<subsystem>/conf/CS.cfg** にある) **CS.cfg** および (**/var/lib/pki/<instance>/alias/** にある) NSS データベースのシステム証明書を確認します。そうでない場合は、認証局 (CA) が起動できません。

- **システム証明書の有効期限**

  インストールされているシステム証明書の有効期限ステータスを確認します (詳細については、System Certificate を参照してください)。

- **システムの証明書信頼フラグ**

  インストールされたシステム証明書に正しい Trust フラグが付いているかどうかを確認します (詳細については、System Certificate を参照してください)。

- **サブシステムの接続チェック**

  サブシステムが実行中かどうかを確認し、要求に応答できるかどうかを確認します。

- **サブシステムのクローン接続およびデータチェック**

  特定の CS サブシステム内で設定された一連のクローンの単純な接続とデータの健全性をチェックします。指定された CA サブシステムのセキュリティードメインは、設定されたクローンを識別するために参照されます。その後、チェックは各クローンにアクセスし、該当する場合はデータの健全性を検証します。

22.2.1.2. PKI Healthcheck 設定

PKI ヘルスチェックツールの設定は **/etc/pki/healthcheck.conf** に保存されます。以下のようになります。

```
[global]
        plugin_timeout=300
        cert_expiration_days=30

        # Dogtag specific section
        [dogtag]
        instance_name=pki-tomcat
```

22.2.1.3. PKI Healthcheck の実行

- ヘルスチェックを実行するには、**pki-healthcheck** コマンドを実行します。
- 特定のチェックを実行することもできます。以下に例を示します。

```
# pki-healthcheck --source pki.server.healthcheck.meta.csconfig --check
DogtagCertsConfigCheck
```

可能なオプションの詳細は、man ページ **man pki-healthcheck** を参照してください。

22.2.1.4. Healthcheck の出力形式

ヘルスチェックでは、以下の出力が生成されます。これは、**output-type** を使用して設定できます。

- デフォルトでは、マシンが判読できる出力 (JSON 形式 **json**)
- または、人間が判読できる出力 (**human**)。

--output-file オプションを使用して、代替ファイルの宛先を指定できます。

22.2.1.5. Healthcheck の結果

レポートは、実行内容とステータスを説明するメッセージで設定されます。Healthcheck の各モジュールは、次のいずれかの結果を返します。

SUCCESS
: 予想どおりに設定され、チェックが実行され、問題が検出されない

警告
: エラーではありませんが、注目または評価する価値があります (証明書はまもなく有効期限が切れます)。

ERROR
: 期待どおりに設定されておらず、何か問題がありますが、サーバーはまだ機能している可能性があります (クローンの競合など)

CRITICAL
: 期待どおりに設定されておらず、影響を受ける可能性が高い (たとえば、サービスが開始されていない、証明書の有効期限が切れているなど)

ステータスが成功しない場合、メッセージには追加情報または推奨事項が含まれている可能性があります。これらは、管理者が問題を修正するために使用できます (たとえば、ファイルのパーミッションが間違っている、X が予期されているが Y が発生したなど)。

パート V. 参考資料

付録A 証明書プロファイルの入力および出力の参照

プロファイルの入力と出力は、証明書要求で予想される入力パラメーターと登録結果の出力形式を定義します。Red Hat Certificate System の他の多くのコンポーネントと同様に、プロファイルの入力と出力は、カスタマイズと柔軟性を提供するために JAVA プラグインとして実装されています。この付録では、デフォルトの入力プラグインおよび出力プラグインのリファレンスを提供します。

A.1. 入力の参照

入力により、特定の証明書プロファイルに関連付けられた登録ページに特定のフィールドが配置されます。証明書プロファイルに設定された入力は、適切なフィールドを使用して動的に登録ページを生成するために使用されます。これらの入力フィールドは、プロファイルが最終的な証明書を生成するために必要な情報を収集します。

A.1.1. CMC 証明書要求入力

Certificate Request 入力は、証明書要求が登録フォームに貼り付けられる登録に使用されます。ドロップダウンリストからリクエスト形式を設定できるようにし、リクエストを貼り付ける入力フィールドを提供します。

この入力により、以下のフィールドが登録フォームに置かれます。

- **証明書要求のタイプ**。このドロップダウンメニューにより、ユーザーが証明書要求のタイプを指定できます。PKCS #10 または CRMF を選択できます。Cryptographic Message Syntax (CMC) 登録を介した Certificate Management Message は、PKCS #10 と CRMF の両方でサポートされています。
- **証明書要求**。このテキストエリアで、リクエストを貼り付けます。

例A.1

```
caAdminCert.cfg:input.i1.class_id=certReqInputImpl
```

A.1.2. CMC 証明書要求入力

CMC Certificate Request 入力は、Certificate Message over CMS (CMC) 証明書要求を使用した登録に使用され、要求フォームで送信されます。要求タイプは PKCS #10 または CRMF のいずれかである必要があり、唯一のフィールドは要求を貼り付けるための **Certificate Request** テキスト領域です。

例A.2

```
caCMCUserCert.cfg:input.i1.class_id=cmcCertReqInputImpl
```

A.1.3. デュアルキー生成入力

デュアルキー生成入力は、デュアルキーペアが生成される登録用であり、したがって、署名用と暗号化用の 2 つの証明書が発行されます。

この入力により、以下のフィールドが登録フォームに置かれます。

- **Key Generation Request Type**。このフィールドは、リクエストタイプ **crmf** として表示される読み取り専用フィールドです。
- **キー生成要求**。このフィールドは、暗号化証明書と署名証明書の両方のキー生成要求でのキーサイズの選択を設定します。

例A.3

```
caDualCert.cfg:input.i1.class_id=dualKeyGenInputImpl
```

A.1.4. ファイル署名入力

File-Signing 入力は、ファイルが改ざんされていないことを示すためにファイルに署名するフィールドを設定します。

この入力により、以下のフィールドが作成されます。

- **Key Generation Request Type**。このフィールドは、リクエストタイプ **crmf** として表示される読み取り専用フィールドです。
- **キー生成要求**。この入力でドロップダウンメニューが追加され、キー生成要求で使用する鍵のサイズを選択します。
- **URL Of File Being Signed**。これにより、署名されるファイルの場所が指定されます。
- **Text Being Signed**。これでファイル名が指定されます。

例A.4

```
caAgentFileSigning.cfg:input.i2.class_id=fileSigningInputImpl
```

A.1.5. イメージ入力

Image 入力は、イメージファイルに署名するフィールドを設定します。この入力が作成する唯一のフィールドは **Image URL** で、署名されるイメージの場所を示すイメージを示します。

A.1.6. キー生成入力

キー生成入力は、単一のキーペアが生成される登録 (通常はユーザーベースの証明書登録) に使用されます。

この入力により、以下のフィールドが登録フォームに置かれます。

- **Key Generation Request Type**。このフィールドは、リクエストタイプ **crmf** として表示される読み取り専用フィールドです。

- **キー生成要求**。この入力でドロップダウンメニューが追加され、キー生成要求で使用する鍵のサイズを選択します。

例A.5

```
caDualCert.cfg:input.i1.class_id=keyGenInputImpl
```

A.1.7. nsHKeyCertRequest (Token Key) Input

Token Key 入力は、エージェントが後で証明書ベースの認証に使用するハードウェアトークンのキーを登録するために使用されます。

この入力により、以下のフィールドが登録フォームに置かれます。

- **トークンキー CUID**。このフィールドは、トークンデバイスの CUID (通常は一意のユーザー ID) を入力します。
- **Token Key User Public Key**。このフィールドには、トークンユーザーの公開鍵が含まれている必要があります。

例A.6

```
caTempTokenDeviceKeyEnrollment.cfg:input.i1.class_id=nsHKeyCertReqInputImpl
```

A.1.8. nsNKeyCertRequest (トークンユーザーキー) 入力

Token User Key 入力は、ハードウェアトークンのユーザーのキーを登録するために使用され、エージェントは後で証明書ベースの認証にトークンを使用します。この入力により、以下のフィールドが登録フォームに置かれます。

- **トークンキーユーザー UID**。このフィールドは、トークンデバイスのユーザーの LDAP エントリーの UID を指定します。
- **Token Key User Public Key**。このフィールドには、トークンユーザーの公開鍵が含まれている必要があります。

例A.7

```
caTempTokenUserEncryptionKeyEnrollment.cfg:input.i1.class_id=nsNKeyCertReqInputImpl
```

A.1.9. Serial Number Renewal 入力

Serial Number Renewal 入力は、CA が元の証明書エントリーを取得し、その情報を使用して証明書を再生成できるように、既存の証明書のシリアル番号を設定するために使用されます。この入力により、**Serial Number** フィールドが登録フォームに挿入されます。

これは、更新フォームで使用する必要がある唯一の入力です。他のすべての情報は、証明書エントリーによって提供されます。

例A.8

```
caTokenUserEncryptionKeyRenewal.cfg:input.i1.class_id=serialNumRenewInputImpl
```

A.1.10. 発行先の DN 入力

Subject DN 入力を使用すると、ユーザーは特定の DN を入力して証明書のサブジェクト名として設定でき、入力によって 1 つの **Subject Name** フィールドが登録フォームに挿入されます。

例A.9

```
caAdminCert.cfg:input.i3.class_id=subjectDNInputImpl
```

A.1.11. サブジェクト名入力

サブジェクト名の入力は、DN パラメーターをユーザーから収集する必要がある場合の登録に使用されます。パラメーターは、証明書のサブジェクト名を形成するために使用されます。この入力により、以下のフィールドが登録フォームに置かれます。

- **UID** (LDAP ディレクトリーのユーザー ID)
- **E メール**
- **Common Name** (ユーザー名)
- **Organizational Unit** (ユーザーが属する組織単位 (***ou***)
- **Organization** (組織名)
- **国** (ユーザーが置かれている国)

例A.10

```
caDualCert.cfg:input.i2.class_id=subjectNameInputImpl
```

A.1.12. 送信元情報の入力

Submitter Information 入力は、名前、電子メール、電話番号などの証明書要求者の情報を収集します。

この入力により、以下のフィールドが登録フォームに置かれます。

- Requester Name
- Requester Email
- Requester Phone

例A.11

```
caAdminCert.cfg:input.i2.class_id=submitterInfoInputImpl
```

A.1.13. 一般的な入力

Generic Input を使用すると、管理者は、パターンを処理する拡張プラグインで使用する入力フィールドをいくつでも指定できます。たとえば、**ccm** パラメーターおよび **GUID** パラメーターは、パターン化されたサ Subject Alternative Name Extension Default プラグインで使用されます。

例A.12

```
input.i3.class_id=genericInputImpl
input.i3.params.gi_display_name0=ccm
input.i3.params.gi_param_enable0=true
input.i3.params.gi_param_name0=ccm
input.i3.params.gi_display_name1=GUID
input.i3.params.gi_param_enable1=true
input.i3.params.gi_param_name1=GUID
input.i3.params.gi_num=2
…
policyset.set1.p6.default.class_id=subjectAltNameExtDefaultImpl
policyset.set1.p6.default.name=Subject Alternative Name Extension Default
policyset.set1.p6.default.params.subjAltExtGNEnable_0=true
policyset.set1.p6.default.params.subjAltExtGNEnable_1=true
policyset.set1.p6.default.params.subjAltExtPattern_0=$request.ccm$
policyset.set1.p6.default.params.subjAltExtType_0=DNSName
policyset.set1.p6.default.params.subjAltExtPattern_1=
(Any)1.3.6.1.4.1.311.25.1,0410$request.GUID$
policyset.set1.p6.default.params.subjAltExtType_1=OtherName
policyset.set1.p6.default.params.subjAltNameExtCritical=false
policyset.set1.p6.default.params.subjAltNameNumGNs=2
```

A.1.14. Subject Alternative Name Extension 入力

Subject Alternative Name Extension 入力は、Subject Alternative Name Extension Default プラグインとともに使用されます。これにより、管理者は、入力へのパターン **req_san_pattern_#** を使用して URI の番号付きパラメーター、**SubjectAltNameExt** 拡張を有効にできます。たとえば、以下が含まれる URI などです。

```
...&req_san_pattern_0=host0.Example.com&req_san_pattern_1=host1.Example.com
```

host0.Example.com および **host1.Example.com** を以下のプロファイルから **SubjectAltNameExt** 拡張機能に挿入します。

例A.13

```
input.i3.class_id=subjectAltNameExtInputImpl
input.i3.name=subjectAltNameExtInputImpl
…
policyset.serverCertSet.9.constraint.class_id=noConstraintImpl
policyset.serverCertSet.9.constraint.name=No Constraint
policyset.serverCertSet.9.default.class_id=subjectAltNameExtDefaultImpl
```

```
policyset.serverCertSet.9.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_0=$request.req_san_pattern_0$
policyset.serverCertSet.9.default.params.subjAltExtType_0=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.req_san_pattern_1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=$request.req_san_pattern_2$
policyset.serverCertSet.9.default.params.subjAltExtType_2=DNSName
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=3
```

A.2. 出力の参照

出力は、登録に成功したエンドユーザーへの応答です。

A.2.1. 証明書の出力

この出力では、証明書が pretty-print 形式で表示されます。この出力は設定または変更できません。これは、証明書以外のものは pretty-print 形式では表示されません。

自動登録には、この出力を指定する必要があります。ユーザーが自動登録方法を使用して正常に認証されると、証明書が自動的に生成され、この出力ページがユーザーに返されます。エージェントが承認した登録では、ユーザーは、証明書が発行されると、エンドエンティティーページで要求 ID を指定することにより、証明書を取得できます。

例A.14

```
caAdminCert.cfg:output.o1.class_id=certOutputImpl
```

A.2.2. PKCS #7 出力

この出力は、証明書と証明書チェーンを PKCS #7 形式で返します。PKCS #7 形式は、署名に使用される Cryptographic Message Syntax Standard です。この出力は設定または変更できません。

例A.15

```
caAgentFileSigning.cfg:output.o1.class_id=pkcs7OutputImpl
```

A.2.3. nsNSKeyOutput

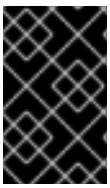
このクラスは、トークンキーの DER エンコードされた証明書を返す出力プラグインを実装します。

A.2.4. CMMF 出力

この出力は、CMMF (Certificate Management Messages Formats) で証明書を返します。CMMF は、PKI のさまざまな部分間の通信を管理し、証明書の要求と証明書の失効の要求に使用されます。

付録B 証明書および CRL のデフォルト、制約、および拡張

この付録では、X.509 v3 で定義された標準の証明書拡張と、X.509 v3 が完成する前にリリースされた製品のバージョンで使用された Netscape で定義された拡張機能の両方を説明します。PKIX Part 1 の推奨事項など、特定の種類の証明書で使用する拡張機能の推奨事項を提供します。



重要

この付録は、Red Hat Certificate System で使用または設定可能なデフォルト、制約、証明書および CRL 拡張機能のリファレンスです。証明書および CRL 拡張の詳細な参照および説明は、RFC 3280 を参照してください。

この付録には以下のセクションが含まれます。

- 「デフォルトの参照」
- 「制約の参照」
- 「標準仕様の X.509 v3 証明書拡張機能リファレンス」
- 「CRL 拡張機能」

B.1. デフォルトの参照

デフォルトでは、証明書の内容の定義に使用されます。このセクションでは、事前定義されたデフォルト値を一覧表示し、定義します。

B.1.1. Authority Info Access 拡張のデフォルト

デフォルトでは、Authority Info Access 拡張をアタッチします。この拡張機能は、証明書を検証するアプリケーションが、証明書を発行した CA に関するオンライン検証サービスや CA ポリシーデータなどの情報にアクセスする方法を指定します。この拡張機能は、CA によって維持されている CRL の場所を直接指すために使用しないでください。CRL Distribution Points 拡張 「CRL Distribution Points 拡張機能のデフォルト」 は、CRL の場所への参照を提供します。

この拡張機能に関する一般的な情報は、「authorityInfoAccess」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.1 Authority Info Access Extension のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
Method_**n**	拡張機能が表示されている証明書を発行した CA に関する追加情報を取得するアクセス方法を指定します。以下の値のいずれかになります。 ● ocsp (1.3.6.1.5.5.7.48.1). ● calssuers (1.3.6.1.5.5.7.48.2) ● renewal (2.16.840.1.113730.16.1)
LocationType_**n**	証明書を発行した CA に関する追加情報を含む場所の一般名タイプを指定します。これは以下のいずれかのタイプになります。 ● DirectoryName ● DNSName ● EDIPartyName ● IPAddress ● OID ● RFC822Name ● URIName

パラメーター	説明
Location_**n**	証明書を発行した CA に関する追加情報を取得するためのアドレスまたは場所を指定します。 ● **directoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **dNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 ● **OID** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。 ● **URIName** の場合、値は、URL 構文およびエンコード規則に従った非相対ユニバーサルリソース識別子 (URI) である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://ocspResponder.example.com:8000**Certificate System は、IPv4 と IPv6 の IP アドレスの両方を許可します。
Enable_**n**	この場所を有効にするかどうかを指定します。**true** を選択してセットとしてマークします。**false** を選択して無効にします。

B.1.2. Authority Key Identifier 拡張機能のデフォルト

デフォルトでは、認証局キー識別子の拡張子が証明書に接続されます。拡張機能は、CA が証明書に署名するために使用する秘密鍵に対応する公開鍵を識別します。このデフォルトにはパラメーターがありません。この拡張を使用すると、公開鍵情報とともに証明書に含まれます。

このデフォルトには、次の制約があります。

- No Constraints は、「No Constraint」 を参照してください。

この拡張機能に関する一般的な情報は、「authorityKeyIdentifier」 を参照してください。

B.1.3. 認証トークンサブジェクト名のデフォルト

このプロファイルのデフォルトでは、認証トークン (AuthToken) オブジェクトの属性値に基づいてサブジェクト名が入力されます。

このデフォルトのプラグインは、ディレクトリーベースの認証マネージャーと動作します。Directory-Based User Dual-Use Certificate Enrollment 証明書登録証明書プロファイルには、UID とパスワードの 2 つの入力パラメーターがあります。ディレクトリーベースの認証マネージャーは、所定の UID とパスワードが正しいかどうかを確認します。

さらに、ディレクトリーベースの認証マネージャーは、発行する証明書のサブジェクト名を作成します。これは、**AuthToken** からのユーザーの DN 値を使用してサブジェクト名を形成します。

このデフォルトは、**AuthToken** からサブジェクト名を読み取り、それを証明書要求に配置して、最終的な証明書にサブジェクト名が含まれるようにするロールを果たします。

次の制約は、このデフォルトで定義できます。

- No Constraints は、「No Constraint」 を参照してください。

B.1.4. 基本的な制約拡張機能のデフォルト

デフォルトでは、基本制約の拡張を証明書にアタッチします。拡張機能は、証明書マネージャーが CA であるかどうかを特定します。この拡張機能は、証明書チェーンの検証プロセス中に、CA 証明書を識別し、証明書チェーンパスの長さの制約を適用するためにも使用されます。

この拡張機能に関する一般的な情報は、「basicConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Basic Constraints 拡張機能制約は、「Basic Constraints 拡張機能制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.2 基本的な制約エクステンションのデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
IsCA	証明書サブジェクトが CA であるかどうかを指定します。**true** の場合、サーバーは **PathLen** パラメーターをチェックして、証明書に指定したパスの長さを設定します。**false** の場合、サーバーは証明書のサブジェクトを CA 以外として処理し、**PathLen** パラメーターに指定された値を無視します。
PathLen	パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、**maxPathLen** パラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraint 拡張で指定されたパスの長さよりも短くする必要があります。**0** は、従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。**n** は、ゼロよりも大きい整数でなければなりません。従属 CA 証明書の下で許可される従属 CA 証明書の最大数を指定します。 フィールドが空白の場合、パスの長さはデフォルトで、発行者の証明書の Basic Constraint 拡張機能で設定されたパスの長さによって決定されます。発行者のパスの長さが無制限の場合は、下位 CA 証明書のパスの長さも無制限になります。発行者のパス長がゼロより大きい整数の場合、下位 CA 証明書のパス長は、発行者のパス長より 1 小さい値に設定されます。たとえば、発行者のパス長が 4 の場合、下位 CA 証明書のパス長は 3 に設定されます。

B.1.5. CA 有効性のデフォルト

このデフォルトでは、CA 証明書の登録または更新プロファイルにオプションが追加され、CA の署名証明書の有効期限の制約がバイパスされます。これは、発行された CA 証明書の有効期限が、発行された C A 署名証明書の有効期限よりも遅い可能性があることを意味します。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.3 CA 有効性のデフォルトパラメーター

パラメーター	説明

パラメーター	説明
bypassCAnotafterrange	要求側の CA が、発行側の CA の有効期間を超えて有効期間が延長された証明書を要求できるかどうかのデフォルト値を設定します。
range	この証明書の絶対有効期間を日数で指定します。
startTime	現在の時間に基づいて有効期間が始まるタイミングを設定します。

B.1.6. 証明書ポリシーの拡張機能のデフォルト

デフォルトでは、Certificate Policy Mappings の拡張を証明書テンプレートに割り当てます。この拡張機能は、証明書が発行されたポリシーと証明書を使用できる目的を示す 1 つ以上のポリシーを定義します。デフォルトでは、最大 5 つのポリシーが定義されますが、値を変更できます。

この拡張機能に関する一般的な情報は、「certificatePoliciesExt」 を参照してください。

表B.4 証明書ポリシー拡張のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
numCertPolicies	定義できるポリシーの数を指定します。デフォルトは **5** です。
enable	**true** を選択してポリシーを有効にします。**false** を選択してポリシーを無効にします。
policyId	ポリシーの OID 識別子を指定します。
cpsURI.enable	拡張機能には、発行者 Certificate Practice Statement への URI を含めることができます。**true** を選択して URI を有効にします。**false** を選択して URI を無効にします。
CPSURI.value	この値は、CA によって公開される Certification Practice Statement (CPS) へのポインターです。ポインターは URI の形式になります。
usernotice.enable	拡張機能には、発行者の Certificate Practice Statement への URI を含めることも、ユーザー通知などの発行者情報をテキスト形式で埋め込むこともできます。ユーザー通知を有効にするには **true** を選択します。ユーザー通知を無効にするには、**false** を選択します。

パラメーター	説明
usernotice.noticeReference.noticeNumbers	この任意のユーザー通知パラメーターは、他の場所に保存されているメッセージを指す一連の番号です。
usernotice.noticeReference.organization	このオプションのユーザー通知パラメーターは会社名を指定します。
usernotice.explicitText.value	この任意のユーザー通知パラメーターには、証明書内のメッセージが含まれます。

B.1.7. CRL Distribution Points 拡張機能のデフォルト

デフォルトでは、CRL Distribution Points の拡張を証明書に割り当てます。この拡張機能は、証明書を検証しているアプリケーションが CRL 情報を取得して、証明書の失効ステータスを検証できる場所を識別します。

この拡張機能に関する一般的な情報は、「CRLDistributionPoints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.5 CRL Distribution Points 拡張設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
Type_**n**	CRL ディストリビューションポイントのタイプを指定します。許容値は **DirectoryName**、**URIName**、または **RelativeToIssuer** です。型は、**Name** フィールドの値に対応する必要があります。

パラメーター	説明
Name_**n**	CRL 配布ポイントの名前を指定します。名前は次のいずれかの形式にすることができます。 ● RFC 2253 構文の X.500 ディレクトリー名。名前は、**cn=CA Central, ou=Research Dept, o=Example Corporation, c=US** のように、証明書のサブジェクト名に似ています。 ● URIName (**http://testCA.example.com:80** など)。 ● CRL 発行者に対して相対的な場所を指定する RDN。この場合、**Type** 属性の値は **RelativeToIssuer** である必要があります。
Reasons_**n**	配布ポイントで保持される CRL で想定される失効理由を指定します。次の定数のコンマ区切りリストを提供します。 ● unused ● keyCompromise ● cACompromise ● affiliationChanged ● superseded ● cessationOfOperation ● certificateHold
IssuerType_**n**	ディストリビューション中に保持される CRL を署名した発行者の命名タイプを指定します。発行者名は以下のいずれかの形式になります。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName
IssuerName_**n**	CRL に署名した CRL 発行者の名前を指定します。許容値は次のとおりです。

パラメーター	説明
	● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 ● **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 ● **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample** このパラメーターの値は、**issuerName** フィールドの値に対応している必要があります。

パラメーター	説明

B.1.8. Extended Key Usage 拡張機能のデフォルト

デフォルトでは、Extended Key Usage の拡張を証明書に登録します。

この拡張機能に関する一般的な情報は、「extKeyUsage」 を参照してください。

この拡張機能は、Key Usage 拡張機能に示されている基本的な目的に加えて、認証された公開鍵を使用できる目的を識別します。たとえば、Extended Key Usage 拡張が署名キーを特定した場合、Extended Key Usage の拡張では、キーの使用法を OCSP 応答のみに署名したり、Java™ アプレットだけに署名するのに絞り込むことができます。

表B.6 Extended Key Usage 拡張機能の PKIX 使用定義

使用方法	OID
サーバー認証	1.3.6.1.5.5.7.3.1
クライアント認証	1.3.6.1.5.5.7.3.2
コード署名	1.3.6.1.5.5.7.3.3
Email	1.3.6.1.5.5.7.3.4
IPsec エンドシステム	1.3.6.1.5.5.7.3.5
IPsec トンネル	1.3.6.1.5.5.7.3.6
IPsec ユーザー	1.3.6.1.5.5.7.3.7
タイムスタンプ	1.3.6.1.5.5.7.3.8

Windows 2000 は、暗号化されたファイルシステム (EFS) と呼ばれる機能を使用して、ハードディスク上のファイルを暗号化できます。以下の 2 つの OID を持つ Extended Key Usage が含まれる証明書を使用します。

1.3.6.1.4.1.311.10.3.4 (EFS 証明書)

1.3.6.1.4.1.311.10.3.4.1 (EFS リカバリー証明書)

EFS リカバリー回復証明書は、ユーザーが秘密鍵を紛失し、その鍵で暗号化されたデータを使用する必要がある場合に、復元エージェントによって使用されます。Certificate System は、これら 2 つの OID をサポートし、これらの OID を含む Extended Key Usage 拡張機能を含む証明書を発行できるようにします。

通常のユーザー証明書は、リカバリー OID ではなく、EFS OID のみで作成する必要があります。

次の制約は、このデフォルトで定義できます。

- 拡張鍵の使用に関する制約。「拡張された鍵使用拡張制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。

- No Constraints は、「No Constraint」 を参照してください。

表B.7 Extended Key Usage 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
OID	キー使用目的を識別する OID を指定します。許容値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID です。たとえば、**2.16.840.1.113730.1.99** です。キーの使用目的に応じて、OID は PKIX (表B.6「Extended Key Usage 拡張機能の PKIX 使用定義」 にリストされている) またはカスタム OID で指定できます。カスタム OID は、会社で使用するために予約された ID の登録済みサブツリーである必要があります。 Certificate System の評価とテストにカスタム OID を使用することは可能ですが、実稼働環境では、OID の定義と ID のサブツリーの登録に関する ISO 規則に準拠しています。

B.1.9. Freshest CRL 拡張機能のデフォルト

デフォルトでは、Freshest CRL 拡張を証明書に割り当てます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.8 Freshest CRL 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
PointEnable_**n**	**true** を選択してこのポイントを有効にします。**false** を選択してこのポイント を無効にします。
PointType_**n**	**DirectoryName** または **URIName** のいずれかの発行ポイントのタイプを指定します。

パラメーター	説明
PointName_**n**	• **pointType** が **directoryName** に設定されている場合、この値は証明書のサブジェクト名と同様に X.500 名である必要があります。たとえば、**cn=CACentral,ou=Research Dept,o=Example Corporation,c=US** となります。 • **pointType** が **URIName** に設定されている場合、名前は URI であるホストを指定する絶対パス名である必要があります。例: **http://testCA.example.com/get/crls/here/**。
PointIssuerName_**n**	CRL に署名した発行者の名前を指定します。名前は以下のいずれかの形式になります。 • **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 • **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** • **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** • **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 • **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 • **iPAddress** の場合、値は有効な IP アドレスでなければなりません。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFF**

パラメーター	説明
	F:FFFF:255.255.255.0、および FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000 になります。 • **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、1.2.3.4.55.6.5.99 です。 • **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample** name の値は、**PointType_** で指定した形式に準拠する必要があります。
PointType_**n**	CRL に署名した CRL 発行者の一般的な名前タイプを指定します。許容値は次のとおりです。 • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • IPAddress • OIDName • OtherName このパラメーターの値は、**PointIssuerName** フィールドの値に対応している必要があります。

B.1.10. 一般的な拡張機能のデフォルト

この拡張により、ユーザーが決定したデータで汎用拡張を作成できます。デフォルトでは、汎用拡張が正しく設定されます。

表B.9 一般的な拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
genericExtOID	拡張 OID 識別子を指定します。
genericExtData	拡張に含まれるバイナリーデータ。

B.1.11. Inhibit Any-Policy 拡張機能のデフォルト

CA に発行される証明書には、inhibit any-policy 拡張を使用できます。禁止ポリシー拡張は、値が { 2 5 29 32 0 } の特別な anyPolicy OID が、他の証明書ポリシーに対する明示的な一致とは見なされていないことを示します。

表B.10 Inhibit Any-Policy 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	**このポリシーは Critical とマークする必要があります。** この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
SkipCerts	このパラメーターで any-policy ポリシーが許可されなくなる前に、パスに表示される追加証明書の数を示します。**1** の値は、any-policy は、この証明書のサブジェクトによって発行された証明書で処理できますが、パス内の追加の証明書では処理できないことを示します。

B.1.12. Issuer Alternative Name 拡張機能のデフォルト

このデフォルトは、Issuer Alternative Name 拡張を証明書に割り当てます。Issuer Alternative Name 拡張は、インターネットスタイルのアイデンティティーを証明書発行者に関連付けるために使用されます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトは、各場所のパラメーターを指定して最大 5 つの場所を定義できます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.11 Issuer Alternative Name 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
issuerAltExtType	これにより、使用する名前拡張のタイプが設定されます。これは以下のいずれかになります。 • RFC822Name • DirectoryName • DNSName • EDIPartyName • URIName • IPAddress • OIDName
issuerAltExtPattern	拡張に追加する要求属性値を指定します。属性の値は、サポートされる一般名のタイプに準拠する必要があります。許容値は、証明書要求に含まれる要求属性です。 サーバーがリクエストの属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Issuer Alternative Name 拡張を証明書に追加しません。リクエストから適切な属性を使用して issuerAlternativeName を形成することができない場合は、トークン式なしでリテラル文字列を使用できます。たとえば、**認証局** です。

B.1.13. Key Usage 拡張機能のデフォルト

デフォルトでは、Key Usage 拡張機能が証明書に接続されます。拡張機能は、データ署名、キー暗号化、データ暗号化など、証明書に含まれるキーを使用する目的を指定します。これにより、キーペアの使用が所定の目的に制限されます。

この拡張機能に関する一般的な情報は、「keyUsage」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Key Usage 制約については、「主な使用拡張機能の制約」 を参照してください。
- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.12 Key Usage 拡張機能のデフォルト設定パラメーター

パラメーター	説明

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
digitalSignature	SSL クライアント証明書と S/MIME 署名証明書を許可するかどうかを指定します。 **t rue** を選択して設定します。
nonRepudiation	S/MIME 署名証明書に使用するかどうかを指定します。 **t rue** を選択して設定します。 **警告** このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。
keyEncipherment	サブジェクトの公開鍵を使用して秘密鍵と秘密鍵のどちらを暗号化するかを指定します。これは、SSL サーバー証明書および S/MIME 暗号化証明書に設定されます。 **t rue** を選択して設定します。
dataEncipherment	サブジェクトの公開鍵を使用して、キー資料とは対照的に、拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
keyAgreement	サブジェクトの公開鍵がキー合意に使用されるたびに拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
keyCertsign	公開鍵を使用して他の証明書の署名を検証するかどうかを指定します。この設定は CA 証明書に使用されます。**true** を選択してオプションを設定します。
cRLSign	CRL に署名する CA 署名証明書の拡張を設定するかどうかを指定します。 **t rue** を選択して設定します。
encipherOnly	公開鍵が鍵共有の実行中にデータを暗号化するためだけのものである場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合、**keyAgreement** も設定する必要があります。 **t rue** を選択して設定します。

パラメーター	説明
decipherOnly	公開鍵が鍵共有の実行中にデータを暗号化するためだけのものである場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合、**keyAgreement** も設定する必要があります。**true** を選択して設定します。

B.1.14. Name Constraints 拡張機能のデフォルト

このデフォルトでは、Name Constraints 拡張を証明書に割り当てます。この拡張機能は CA 証明書で使用され、証明書チェーン内の後続の証明書のサブジェクト名またはサブジェクト代替名を配置するネームスペースを示します。

この拡張機能に関する一般的な情報は、「nameConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

このデフォルトでは、許可されたサブツリーと除外されたサブツリーの両方に最大 5 つの場所が定義され、場所ごとにパラメーターが設定されます。パラメーターには、パラメーターが関連付けられる場所を表示するために、表で **n** のマークが付いています。

表B.13 Name Constraints 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
PermittedSubtrees**n**.min	許可されるサブツリーの最小数を指定します。 - **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 - **0** は、サブツリーの最小数がゼロであることを指定します。 - **n** は、ゼロより大きい整数である必要があります。サブツリーの最小数を設定します。

パラメーター	説明
PermittedSubtreesmax_**n**	許可されるサブツリーの最大数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最大数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。許可されるサブツリーの最大数を設定します。
PermittedSubtreeNameChoice_**n**	拡張に含める許可されるサブツリーの一般な名前タイプを指定します。許容値は次のとおりです。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName
PermittedSubtreeNameValue_**n**	拡張に含める許可されるサブツリーの汎用名を指定します。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 ● **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名の文字列形式である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める

パラメーター	説明
	必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 ● **IPAddress** の場合、Classless Inter-Domain Routing (CIDR) 表記に準拠する有効な IP アドレスを指定する必要があります。IPv4 アドレスは、**n.n.n.n** 形式、またはネットマスクを使用した **n.n.n.n/m** である必要があります (**10.34.3.133** または **110.34.3.133/24** など)。IPv6 アドレスも CIDR 表記に準拠する必要があります。ネットマスクには、**2620:52:0:2203:527b:9dff:fe56:4495/64** または **2001:db8::/64** があります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **OtherName** は他の形式の名前に使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの **Realm|NameType|NameStrings** 形式になります。 **OtherName** の形式は **(type)oid,string** にする必要があります。例: **(IA5String)1.2.3.4,MyExample**
PermittedSubtreeEnable_**n**	**true** を選択して、このサブツリーエントリーを許可します。
ExcludedSubtrees**n**.min	除外されたサブツリーの最小数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最小数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。これにより、必要なサブツリーの最小数が設定されます。

パラメーター	説明
ExcludedSubtreeMax_**n**	除外されたサブツリーの最大数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、サブツリーの最大数がゼロであることを指定します。 ● **n** は、ゼロより大きい整数である必要があります。これにより、許可されるサブツリーの最大数が設定されます。
ExcludedSubtreeNameChoice_**n**	拡張に追加する除外されたサブツリーの一般名を指定します。許容値は次のとおりです。 ● RFC822Name ● DirectoryName ● DNSName ● EDIPartyName ● URIName ● IPAddress ● OIDName ● OtherName

パラメーター	説明
ExcludedSubtreeNameValue_**n**	拡張に含める許可されるサブツリーの汎用名を指定します。 ● **RFC822Name** の場合、値は有効なインターネットメールアドレスである必要があります。例: **testCA@example.com**。 ● **DirectoryName** の場合は、証明書のサブジェクト名と同様に、値は X.500 名である必要があります。例: **cn=SubCA, ou=Research Dept, o=Example Corporation, c=US** ● **DNSName** の場合、値は有効な完全修飾ドメイン名である必要があります。例: **testCA.example.com** ● **EDIPartyName** の場合、値は IA5String である必要があります。例: **Example Corporation**。 ● **URIName** の場合、値は URL 構文およびエンコーディングルールに続く非相対的な URI である必要があります。名前には、**http** などのスキームと、ホストの完全修飾ドメイン名または IP アドレスを含める必要があります。例: **http://testCA.example.com**。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 ● **IPAddress** の場合、Classless Inter-Domain Routing (CIDR) 表記に準拠する有効な IP アドレスを指定する必要があります。IPv4 アドレスは、**n.n.n.n** 形式、またはネットマスクを使用した **n.n.n.n/m** である必要があります (**10.34.3.133** または **110.34.3.133/24** など)。IPv6 アドレスも CIDR 表記に準拠する必要があります。ネットマスクには、**2620:52:0:2203:527b:9dff:fe56:4495 /64** または **2001:db8::/64** があります。 ● **OIDName** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。 ● **OtherName** の場合、値は他の形式の名前です。これは、**PrintableString**、**IA5String**、**UTF8 String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample**

パラメーター	説明
ExcludedSubtreeEnable_n	**true** を選択して、この除外されたサブツリーエントリーを有効にします。

B.1.15. Netscape Certificate Type 拡張機能のデフォルト

警告

この拡張機能は廃止されています。代わりに、Key Usage または Extended Key Usage による証明書拡張を使用してください。

デフォルトでは、Netscape Certificate Type 拡張を証明書に割り当てます。拡張機能は、CA 証明書、サーバー SS L 証明書、クライアント SSL 証明書、S/MIME 証明書などの証明書タイプを識別します。これにより、証明書の使用が事前に決定された目的に制限されます。

B.1.16. Netscape Comment 拡張機能のデフォルト

警告

この拡張機能は廃止されています。

デフォルトでは、Netscape Comment 拡張機能が証明書にアタッチされます。拡張機能を使用して、証明書にテキスト形式のコメントを含めることができます。コメントを解釈できるアプリケーションは、証明書が使用または表示されるときにコメントを表示します。

この拡張機能に関する一般的な情報は、「netscape-comment」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.14 Netscape Comment 拡張機能の設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
CommentContent	証明書に表示するコメントの内容を指定します。

B.1.17. デフォルト拡張機能なし

デフォルトを使用しない場合は、このデフォルトを使用して制約を設定できます。このデフォルトには設定がなく、デフォルトも設定されていませんが、使用可能なすべての制約を設定できます。

B.1.18. OCSP No Check 機能拡張のデフォルト

デフォルトでは、OCSP No Check 拡張機能が証明書に割り当てられます。拡張機能は、OCSP レスポンダー証明書でのみ使用する必要があり、OCSP 準拠のアプリケーションが、承認された OCSP レスポンダーが OCSP 応答に署名するために使用する証明書の失効ステータスを確認する方法を示します。

この拡張機能に関する一般的な情報は、「OCSPNocheck」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.15 OCSP No Check 機能拡張のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

B.1.19. Policy Constraints 拡張機能のデフォルト

このデフォルトでは、Policy Constraints 拡張を証明書に割り当てます。拡張機能は CA 証明書でのみ使用でき、パス検証を 2 つの方法で制限します。ポリシーマッピングを禁止するか、パス内の各証明書に受け入れ可能なポリシー識別子が含まれていることを要求します。デフォルトでは、**Req ExplicitPolicy** と **InhibitPolicy Mapping** の両方を指定できます。PKIX 標準では、証明書に存在する場合、拡張子が null シーケンスで設定されてはならないことが要求されています。少なくとも 2 つの指定されたフィールドが存在する必要があります。

この拡張機能に関する一般的な情報は、「policyConstraints」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.16 Policy Constraints 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。

パラメーター	説明
reqExplicitPolicy	明示的なポリシーが必要になる前に、パスで許可される証明書の総数を指定します。これは、受け入れ可能なポリシーが必要になる前に、下位 CA 証明書の下にチェーンできる CA 証明書の数です。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、明示的なポリシーが必要になる前に、パスで従属 CA 証明書を許可しないことを指定します。 ● **n** は、ゼロより大きい整数である必要があります。明示的なポリシーが必要になる前に、パスで許可される従属 CA 証明書の最大数を指定します。 この数は、証明書の検証中に使用される CA 証明書の数に影響します。このチェーンは、チェーンを検証して移動させるエンドエンティティー証明書で始まります。このパラメーターは、拡張がエンドエンティティーの証明書に設定されている場合は有効ではありません。
inhibitPolicyMapping	ポリシーマッピングが許可されなくなる前に、パスで許可される証明書の総数を指定します。 ● **-1** は、拡張機能でフィールドを設定すべきではないことを指定します。 ● **0** は、ポリシーマッピングが許可されなくなる前に、パスで従属 CA 証明書が許可されないことを指定します。 ● **n** は、ゼロより大きい整数である必要があります。ポリシーマッピングが許可されなくなる前に、パスで許可される従属 CA 証明書の最大数を指定します。たとえば、値を 1 にすると、ポリシーマッピングは、この証明書のサブジェクトによって発行された証明書で処理できますが、パス内の追加の証明書では処理できないことを示します。

B.1.20. Policy Mappers 拡張機能のデフォルト

このデフォルトでは、Policy Mappings の拡張を証明書にアタッチします。拡張機能は OID のペアを一覧表示し、それぞれのペアが 2 つの CA のポリシーステートメントを識別します。ペアリングは、ある CA の対応するポリシーが別の CA のポリシーと同等であることを示します。この拡張は、クロス証明書のコンテキストで役に立ちます。サポートされる場合、拡張は CA 証明書のみに含まれます。デフォルトでは、ポリシーステートメントに割り当てられた OID をペアにすることにより、ある CA のポリシーステートメントを別の CA のポリシーステートメントにマップします。

各ペアは、**issuerDomainPolicy** と **subjectDomainPolicy** の 2 つのパラメーターで定義されます。ペアは、発行した CA が、サブジェクト CA の **subjectDomainPolicy** と同等の **issuerDomainPolicy** を考慮することを意味します。CA の発行元のユーザーは、特定のアプリケーションの

issuerDomainPolicy を受け入れる可能性があります。ポリシーマッピングは、サブジェクト CA に関連付けられているどのポリシーが受け入れたポリシーと同等であるかをこれらのユーザーに通知します。

この拡張機能に関する一般的な情報は、「policyMappings」 を参照してください。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.17 Policy Mappers 拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
IssuerDomainPolicy_**n**	別の CA のポリシーステートメントとマッピングするために、発行元 CA のポリシーステートメントに割り当てられた OID を指定します。たとえば、1.2.3.4.5 です。
SubjectDomainPolicy_**n**	発行 CA のポリシーステートメントに対応するサブジェクト CA のポリシーステートメントに割り当てられた OID を指定します。たとえば、6.7.8.9.10 です。

B.1.21. Private Key Usage Period 拡張機能のデフォルト

Private Key Usage Period の拡張機能により、証明書発行者は、証明書自体に秘密鍵に異なる有効期間を指定できます。この拡張は、デジタル署名鍵の使用を目的としています。

表B.18 Private Key Usage Period の設定パラメーター

パラメーター	説明
Critical	この拡張は、常にクリティカルではないはずです。
puStartTime	このパラメーターは、開始時間を設定します。デフォルト値は **0** です。これは、拡張機能がアクティベートされた時点から有効期間を開始します。
puDurationDays	このパラメーターは、使用状況の期間を設定します。デフォルト値は **365** です。これは、拡張機能がアクティブ化されてから 365 日に有効期間を設定します。

B.1.22. 署名アルゴリズムのデフォルト

デフォルトでは、証明書要求に署名アルゴリズムがアタッチされます。このデフォルトは、証明書の署名に使用できる可能なアルゴリズムをエージェントに提示します。

次の制約は、このデフォルトで定義できます。

- アルゴリズム制約の署名は、「アルゴリズム制約の署名」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.19 署名アルゴリズムのデフォルト設定パラメーターの署名

パラメーター	説明
signingAlg	この証明書の作成に使用するデフォルトの署名アルゴリズムを指定します。**signingAlgsAllowed** パラメーターに含まれる値のいずれかを指定すると、エージェントはこの値をオーバーライドすることができます。
signingAlgsAllowed	この証明書の署名に使用できる署名アルゴリズムを指定します。アルゴリズムは以下のいずれか 1 つになります。 • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA

B.1.23. サブジェクト代替名の拡張機能のデフォルト

このデフォルトは、Subject Alternative Name 拡張を証明書に割り当てます。拡張機能は、電子メールアドレス、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、または URI などの追加の ID を証明書のサブジェクトにバインドします。標準では、証明書のサブジェクトフィールドに空のシーケンスが含まれている場合に、Subject Alternative 名の拡張子にサブジェクトの代替名が含まれている必要があり、拡張子にクリティカルのマークが付けられている必要があります。

ディレクトリーベースの認証方法の場合、Certificate System は任意の文字列およびバイト属性の値を取得し、それらを証明書要求に設定できます。これらの属性は、自動登録モジュールで定義された **ldapStringAttributes** および **ldapByteAttributes**フィールドに入力することで設定されます。

認証された属性 (LDAP データベースに格納されている属性を意味する) をこの拡張機能の一部にする必要がある場合は、**$request.X$** トークンの値を使用します。

サブジェクトの代替名にユニバーサル一意識別子 (UUID) を挿入するための追加の属性があります。このオプションは、バージョン 4 UUID の乱数を生成します。パターンは、追加 **subjAltExtSource** パラメーターで番号を生成するサーバーを参照することによって定義されます。

この例では、基本的なサブジェクトの代替名拡張のデフォルトが設定されています。

例B.1 サブジェクト代替名の拡張機能のデフォルト設定

```
policyset.serverCertSet.9.constraint.name=No Constraint
```

```
policyset.serverCertSet.9.default.class_id=subjectAltNameExtDefaultImpl
policyset.serverCertSet.9.default.name=Subject Alternative Name Extension Default
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_0=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_0=$request.requestor_email$
policyset.serverCertSet.9.default.params.subjAltExtType_0=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_1=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_1=$request.SAN1$
policyset.serverCertSet.9.default.params.subjAltExtType_1=DNSName
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_2=true
policyset.serverCertSet.9.default.params.subjAltExtPattern_2=http://www.server.example.com
policyset.serverCertSet.9.default.params.subjAltExtType_2=URIName
policyset.serverCertSet.9.default.params.subjAltExtType_3=OtherName
policyset.serverCertSet.9.default.params.subjAltExtPattern_3=(IA5String)1.2.3.4,$server.source$
policyset.serverCertSet.9.default.params.subjAltExtSource_3=UUID4
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_3=true
policyset.serverCertSet.9.default.params.subjAltExtType_4=RFC822Name
policyset.serverCertSet.9.default.params.subjAltExtGNEnable_4=false
policyset.serverCertSet.9.default.params.subjAltExtPattern_4=
policyset.serverCertSet.9.default.params.subjAltNameExtCritical=false
policyset.serverCertSet.9.default.params.subjAltNameNumGNs=5
```

Subject Alternative Name 拡張機能のデフォルトは、プロファイル属性の証明書要求をチェックします。リクエストに属性が含まれる場合、プロファイルはその値を読み込み、拡張機能に設定します。LDAP ベースの認証が設定されている場合は、Subject Alternative Name 拡張機能のデフォルトで LDAP ディレクトリーから属性値を挿入することもできます。証明書に追加された拡張機能には、設定されたすべての属性が含まれています。

Subject Alternative Name 拡張機能のデフォルトで使用できる変数を、表B.20「サブジェクト代替名に値を挿入する変数」 に記載します。

表B.20 サブジェクト代替名に値を挿入する変数

ポリシーセットトークン	説明
$request.auth_token.cn$	証明書を要求したユーザーの LDAP 共通名 (***cn***) 属性。
$request.auth_token.mail$	証明書を更新したユーザーの LDAP メール (***mail***) 属性の値。
$request.auth_token.tokenCertSubject$	証明書サブジェクト名。
$request.auth_token.uid$	証明書を要求したユーザーの LDAP ユーザー ID (***uid***) 属性。
$request.auth_token.user$	
$request.auth_token.userDN$	証明書を要求したユーザーのユーザー DN。
$request.auth_token.userid$	証明書を要求したユーザーのユーザー ID 属性の値。

ポリシーセットトークン	説明
$request.uid$	証明書を要求したユーザーのユーザー ID 属性の値。
$request.profileRemoteAddr$	要求するユーザーの IP アドレス。これは、クライアントに応じて IPv4 アドレスまたは IPv6 アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。
$request.profileRemoteHost$	ユーザーのマシンのホスト名または IP アドレス。ホスト名は、**http://server.example.com** などの完全修飾ドメイン名およびプロトコルになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。
$request.requestor_email$	要求を送信したユーザーのメールアドレス。
$request.requestowner$	要求を送信した人。
$request.subject$	証明書が発行されるエンティティーのサブジェクト名 DN。たとえば、**uid=jsmith, e=jsmith@example.com** です。
$request.tokencuid$	登録の要求に使用されるスマートカードトークンのカード一意の ID (CUID)。
$request.upn$	Microsoft UPN。これには **(UTF8String)1.3.6.1.4.1.311.20.2.3,$request.upn$** の形式があります。

ポリシーセットトークン	説明
$server.source$	サーバーに対し、サブジェクト名のバージョン 4 の UUID (乱数) コンポーネントを生成するように指示します。この値は常に **(IA5String)1.2.3.4,$server.source$** 形式になります。

1 つのエクステンションに複数の属性を設定できます。**subjAltNameNumGNs** パラメーターは、一覧表示された属性のうち、証明書に追加する必要があるものの数を制御します。このパラメーターはカスタムプロファイルに追加する必要があり、必要な数の属性を含めるためにデフォルトプロファイルで変更する必要がある場合があります。例B.1「サブジェクト代替名の拡張機能のデフォルト設定」で、**subjAltNameNumGNs** が、**5** に設定され、**RFC822Name**、**DNSName**、**URIName**、**OtherName**、および **RFC822Name** 名 (一般名 **_0**、**_1**、**_2**、**_3**、および **_4**) を挿入します。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.21 サブジェクト代替名の拡張機能のデフォルト設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
Pattern	拡張に追加する要求属性値を指定します。属性の値は、サポートされる一般名のタイプに準拠する必要があります。サーバーがリクエストの属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Subject Alternative Name 拡張を証明書に追加しません。許容値は、証明書要求に含まれる要求属性です。たとえば、**$request.requestor_email$** です。
タイプ	request 属性の一般的な名前タイプを指定します。 • request-attribute の値が **local-part@domain** 形式のメールアドレスの場合は **RFC822Name** を選択します。たとえば、**jdoe@example.com** です。 • 証明書のサブジェクト名と同様に、request-attribute 値が X.500 ディレクトリー名の場合は **DirectoryName** を選択します。たとえば、**cn=Jane Doe, ou=Sales Dept, o=Example Corporation, c=US** です。 • request-attribute の値が DNS 名である場合に **DNSName** を選択します。たとえば、**corpDirectory.example.com** です。

パラメーター	説明
	• request-attribute の値が EDI party 名の場合は、**EDIPartyName** を選択します。例: Example Corporation。 • request-attribute 値が、**http** などの両方のスキームを含む非相対 URI である場合、およびホストの完全修飾ドメイン名または IP アドレスの場合は、**URIName** を選択します。例: **http://hr.example.com** です。証明書システムは、IPv4 アドレスと IPv6 アドレスの両方をサポートします。 • request-attribute 値が、ドットで区切られた数値コンポーネント表記で指定された有効な IP アドレスである場合は、**IPAddress** を選択します。たとえば、**128.21.39.40** です。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 • request-attribute 値が、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である場合は、**OIDName** を選択します。たとえば、**1.2.3.4.55.6.5.99** です。 • 他の形式の名前は **OtherName** を選択します。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**KerberosName** には、**realm1|0|userID1,userID2** などの Realm|NameType|NameStrings 形式になります。 **OtherName** の形式は (type)oid,string にする必要があります。例: **(IA5String)1.2.3.4,MyExample**
ソース	ID を生成するために使用する識別ソースまたはプロトコルを指定します。サポートされるソースは UUID4 で、UUID を作成する乱数を生成します。
コンポーネント数 (NumGN)	サブジェクトの別名に含める必要がある名前コンポーネントの数を指定します。

B.1.24. サブジェクトディレクトリー属性の拡張機能のデフォルト

デフォルトでは、Subject Directory 属性の拡張が証明書に割り当てます。Subject Directory Attributes 機能拡張は、証明書の件名に必要なディレクトリー属性の値をすべて伝えます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.22 サブジェクトディレクトリー属性の拡張機能のデフォルトの設定パラメーター

パラメーター	説明
Critical	この拡張機能に critical マークを付けるには **true** を選択してください。noncritical マークを付けるには **false** を選択してください。
名前	属性名。これは、***cn***、***mail*** などの LDAP ディレクトリー属性になります。
Pattern	拡張に追加する要求属性値を指定します。属性値は、属性の許可される値に準拠する必要があります。サーバーが属性を見つけると、拡張に属性値を設定し、その拡張を証明書に追加します。複数の属性が指定され、リクエストに属性が存在しない場合、サーバーは Subject Directory Attributes 拡張を証明書に追加しません。たとえば、**$request.requestor_email$** です。
Enable	その属性が証明書に追加できるかどうかを設定します。**true** を選択して属性を有効にします。

B.1.25. サブジェクト情報アクセス拡張機能のデフォルト

証明書テンプレートに Subject Information Access 拡張機能を設定する登録デフォルトポリシーを実装します。この拡張機能は、拡張機能が表示されている証明書のサブジェクトの情報とサービスにアクセスする方法を示します。

パラメーター	説明
Critical	この拡張はクリティカルではないはずです。
subjInfoAccessNumADs	証明書に含まれる情報アクセスセクションの数。
subjInfoAccessADMethod_n	アクセスメソッドの OID。

パラメーター	説明
subjInfoAccessADMethod_n	アクセスメソッドのタイプ。 • URIName • ディレクトリー名 • DNS 名 • EID パーティー名 • IP アドレス • OID 名 • RFC822Name
subjInfoAccessADLocation_n	タイプ subjInfoAccessADMethod_n を元にした場所 つまり、URI 名の URL。
subjInfoAccessADEnable_n	**true** を選択してこのエクステンションを有効にします。**false** を選択してこのエクステンションを無効にします。

B.1.26. Subject Key Identifier 拡張機能のデフォルト

デフォルトでは、サブジェクトキー識別子の拡張を証明書に割り当てます。この拡張機能は、特定の公開鍵を含む証明書を識別し、同じサブジェクト名を持つ複数の証明書の中から証明書を識別します。

この拡張機能に関する一般的な情報は、「subjectKeyIdentifier」 を参照してください。

有効にすると、拡張機能がまだ存在しない場合、プロファイルは登録要求に Subject Key Identifier Extensio 拡張機能を追加します。CRMF リクエストなど、リクエストに拡張機能が存在する場合は、デフォルトで拡張機能が置き換えられます。エージェントが手動登録要求を承認した後、プロファイルは、すでに存在する Subject Key Identifier Extension を受け入れます。

このデフォルトにはパラメーターがありません。この拡張を使用すると、公開鍵情報とともに証明書に含まれます。

次の制約は、このデフォルトで定義できます。

- Extension Constraint は、「拡張制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.27. サブジェクト名のデフォルト

デフォルトでは、サーバー側の設定可能なサブジェクト名を証明書要求に割り当てます。静的サブジェクト名は、証明書のサブジェクト名として使用されます。

次の制約は、このデフォルトで定義できます。

- Subject Name 制約の場合は、「Subject Name 制約」 を参照してください。

- Unique Subject Name 制約の場合は、「Unique Subject Name 制約」 を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.23 サブジェクト名のデフォルト設定パラメーター

パラメーター	説明
Name	この証明書のサブジェクト名を指定します。

UidPwdDirAuth プラグインから DNPATTERN 値を使用する証明書サブジェクト名を取得する必要がある場合は、Subject Name Default プラグインを使用するようにプロファイルを設定し、以下に示すように、***Name*** パラメーターを、**AuthToken**の SubjectName に置き換えます。

```
policyset.userCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.userCertSet.1.default.name=Subject Name Default
policyset.userCertSet.1.default.params.name=$request.auth_token.tokenCertSubject$
```

B.1.28. ユーザーキーのデフォルト

デフォルトでは、ユーザーが指定したキーを証明書要求に割り当てます。これは必須のデフォルトです。キーは登録要求の一部です。

次の制約は、このデフォルトで定義できます。

- キー制約については、「主要な制約」 を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.29. ユーザー署名アルゴリズムのデフォルト

このデフォルトは、証明書要求にユーザー指定の署名アルゴリズムを設定する登録デフォルトプロファイルを実装します。証明書プロファイルに含まれている場合、これにより、ユーザーは、制約セットに従って、証明書の署名アルゴリズムを選択できます。

署名アルゴリズムの選択肢を登録フォームに追加するための入力は提供されていませんが、この情報を含むリクエストを送信することは可能です。

次の制約は、このデフォルトで定義できます。

- アルゴリズム制約の署名は、「アルゴリズム制約の署名」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

B.1.30. ユーザーのサブジェクト名のデフォルト

デフォルトでは、ユーザーが指定したサブジェクト名を証明書要求に割り当てます。証明書プロファイルに含まれている場合、ユーザーは、設定された制約に従って、証明書のサブジェクト名を指定できます。この拡張機能は、証明書の発行時に元の証明書要求で指定されたサブジェクト名を保持します。

次の制約は、このデフォルトで定義できます。

- Subject Name 制約の場合は、「Subject Name 制約」 を参照してください。

- Unique Subject Name 制約の場合は、「Unique Subject Name 制約」を参照してください。
- No Constraints は、「No Constraint」を参照してください。

B.1.31. ユーザーの有効性のデフォルト

このデフォルトでは、ユーザーが指定した有効性が証明書要求に添付されます。証明書プロファイルに含まれている場合、ユーザーは設定された制約に従って有効期間を指定できます。このデフォルトプロファイルは、証明書が発行されたときに、元の証明書要求でそのユーザー定義の有効期間を保持します。

ユーザー指定の有効期限を登録フォームに追加するための入力は提供されていませんが、この情報を含むリクエストを送信することは可能です。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」を参照してください。

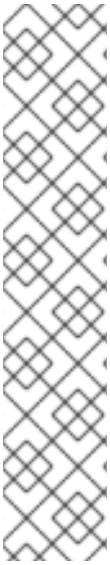
B.1.32. User Supplied Extension Default

User Supplied Extension Default クラスは、証明書要求でユーザーが定義した証明書拡張を証明書に入力します。プロファイルは証明書を登録する前に特定の拡張機能を必要とする可能性があるため、これには、ユーザーが特定の基準を満たす証明書要求を送信するか、特定の情報を提供する必要があります。

警告

この拡張機能のデフォルトの設定には、ユーザーが証明書要求で拡張機能を指定できるため、特に注意してください。このデフォルトを使用する場合、Red Hat は、拡張機能に対応する制約を使用して、User Supplied Extension Default の乱用を最小限に抑えることを強く推奨します。

ユーザー定義の拡張機能は、設定されている制約に対して検証されるため、拡張機能の種類を制限したり (Extension Constraint の制約を介して)、キーやその他の基本的な制約 (CA 証明書かどうかなど) のルールを設定したりできます。



注記

この拡張機能が対応する OID (拡張機能制約) を持つプロファイルに設定されている場合、そのプロファイルを介して処理される証明書要求には、指定された拡張機能を含める **必要があります**。そうでない場合、要求は拒否されます。

エラータ RHSA 2008:0500 の **前** に User Supplied Extension Default で証明書プロファイルが有効になっていた場合は、証明書要求でユーザー提供の拡張機能をサポートするようにこのプロファイルを編集する必要があります。以下の例のように、**userExtensionDefaultImpl** のデフォルトを適用します。指定 OID は、Basic Constraints Extension Constraint に関するものです。

```
policyset.set1.p6.default.class_id=userExtensionDefaultImpl
policyset.set1.p6.default.name=User Supplied Extension Default
policyset.set1.p6.default.params.userExtOID=2.5.29.19
```

CA は、次の 3 つの方法のいずれかで、User Supplied Extension Default を使用した登録を処理します。

- 拡張機能の OID が証明書要求とデフォルトの両方で指定されている場合、拡張機能は制約によって検証され、証明書に適用されます。
- 拡張機能の OID が要求で指定されているが、プロファイルの User Supplied Extension Default で指定されていない場合、ユーザー指定の拡張機能は無視され、証明書はその拡張機能なしで正常に登録されます。
- この拡張機能が対応する OID (拡張機能制約) を持つプロファイルに設定されている場合、そのプロファイルを介して処理される証明書要求には、指定された拡張機能を含める **必要があります**。そうでない場合、要求は拒否されます。

ユーザー定義の拡張を含む証明書 **要求** はプロファイルに送信する必要があります。ただし、証明書登録フォームには、ユーザーが提供する拡張機能を追加するための入力フィールドがありません。拡張機能を提供せずに証明書要求を送信すると失敗します。

例B.2「Extended Key Usage Extension の User Supplied Extension Default」 では、User Supplied Extension Default を、Extended Key Usage Constraint のあるプロファイルに追加します。***userExtOID*** パラメーターで指定された OID は、Extended Key Usage Extension 用です。

例B.2 Extended Key Usage Extension の User Supplied Extension Default

```
policyset.set1.2.constraint.class_id=extendedKeyUsageExtConstraintImpl
policyset.set1.2.constraint.name=Extended Key Usage Extension
policyset.set1.2.constraint.params.exKeyUsageCritical=false
policyset.set1.2.constraint.params.exKeyUsageOIDs=1.3.6.1.5.5.7.3.2,1.3.6.1.5.5.7.3.4
policyset.set1.2.default.class_id=userExtensionDefaultImpl
policyset.set1.2.default.name=User Supplied Extension Default
policyset.set1.2.default.params.userExtOID=2.5.29.37
```

例B.2「Extended Key Usage Extension の User Supplied Extension Default」 では、User Supplied Extension Default により、ユーザーは Extended Key Usage Extension (2.5.29.37) を指定できますが、制約により、ユーザー要求は SSL クライアント認証 (1.3.6.1.5.5.7.3.2) と電子メール保護 (1.3.6.1.5.5.7.3.4) の使用に制限されます。

プロファイルの編集については、「証明書プロファイルの設定」 で説明します。

例B.3 CSR の Multiple User Supplied Extension

RHCS 登録プロファイルフレームワークを使用すると、同じプロファイルで複数の User Supplied Extensions 拡張機能を定義できます。たとえば、以下の組み合わせを指定できます。

- Extended Key Usage Extension の場合:

```
policyset.serverCertSet.2.constraint.class_id=extendedKeyUsageExtConstraintImpl
policyset.serverCertSet.2.constraint.name=Extended Key Usage Extension
policyset.serverCertSet.2.constraint.params.exKeyUsageCritical=false
policyset.serverCertSet.2.constraint.params.exKeyUsageOIDs=1.3.6.1.5.5.7.3.2,1.3.6.1.5.5.7.3.4
policyset.serverCertSet.2.default.class_id=userExtensionDefaultImpl
policyset.serverCertSet.2.default.name=User Supplied Extension Default
policyset.serverCertSet.2.default.params.userExtOID=2.5.29.37
```

- Key Usage Extension の場合:

  以下の形式を使用すると、拡張機能のパラメーターを適用するポリシーを適用できます。

  - CSR: **value = "true" になければなりません**
  - CSR: **value = "false" に存在してはなりません**
  - **オプション**: **value = "-"**

  以下に例を示します。

```
policyset.serverCertSet.13.constraint.class_id=keyUsageExtConstraintImpl
policyset.serverCertSet.13.constraint.name=Key Usage Extension Constraint
policyset.serverCertSet.13.constraint.params.keyUsageCritical=-
policyset.serverCertSet.13.constraint.params.keyUsageCrlSign=false
policyset.serverCertSet.13.constraint.params.keyUsageDataEncipherment=-
policyset.serverCertSet.13.constraint.params.keyUsageDecipherOnly=-
policyset.serverCertSet.13.constraint.params.keyUsageDigitalSignature=-
policyset.serverCertSet.13.constraint.params.keyUsageEncipherOnly=-
policyset.serverCertSet.13.constraint.params.keyUsageKeyAgreement=true
policyset.serverCertSet.13.constraint.params.keyUsageKeyCertSign=-
policyset.serverCertSet.13.constraint.params.keyUsageKeyEncipherment=-
policyset.serverCertSet.13.constraint.params.keyUsageNonRepudiation=-
policyset.serverCertSet.13.default.class_id=userExtensionDefaultImpl
policyset.serverCertSet.13.default.name=User Supplied Key Usage Extension
policyset.serverCertSet.13.default.params.userExtOID=2.5.29.15
```

注記

ユーザー定義の拡張属性で CSR を作成する方法は、「**certutil** を使用したユーザー定義拡張による CSR の作成」 を参照してください。

B.1.33. 有効性のデフォルト

デフォルトでは、サーバー側の設定可能な有効期間を証明書要求に割り当てます。

次の制約は、このデフォルトで定義できます。

- Validity 制約の場合は、「Validity 制約」を参照してください。
- No Constraints は、「No Constraint」 を参照してください。

表B.24 有効性のデフォルト設定パラメーター

パラメーター	説明
range	この証明書の有効期限を指定します。
startTime	現在の時間に基づいて有効期間が始まるタイミングを設定します。

B.2. 制約の参照

制約は、証明書の許容される内容とその内容に関連付けられた値を定義するために使用されます。このセクションでは、それぞれの完全定義を含む事前定義された制約を一覧表示します。

B.2.1. Basic Constraints 拡張機能制約

Basic Constraints 拡張制約は、証明書要求の基本制約がこの制約で設定された基準を満たしているかどうかを確認します。

表B.25 Basic Constraints 拡張機能の制約設定パラメーター

パラメーター	説明
basicConstraintsCritical	エクステンションは critical または noncritical のマークを付けるかどうかを指定します。この拡張機能をクリティカルとしてマークする場合は **true** を選択します。この拡張機能がクリティカルとしてマークされないようにするには、**false** を選択します。ハイフン **-** を選択しても、影響度が重大を意味します。
basicConstraintsIsCA	証明書サブジェクトが CA であるかどうかを指定します。**true** を選択すると、(CA である) このパラメーターに **true** の値を要求します。**false** を選択して、このパラメーターの **true** の値を無効にします。ハイフン **-** を選択すると、このパラメーターに制約を設定しないことを示します。

パラメーター	説明
basicConstraintsMinPathLen	最小許容パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、このパラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraint 拡張で指定されたパスの長さよりも短くする必要があります。 **0** は、発行されている従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。 **n** は、ゼロよりも大きい整数でなければなりません。これは、使用されている従属 CA 証明書の下で許可される従属 CA 証明書の最小数です。
basicConstraintsMaxPathLen	最大許容パスの長さ、つまり発行されている従属 CA 証明書の下 (従属) にチェーンできる CA 証明書の最大数を指定します。パスの長さは、証明書の検証時に使用する CA 証明書の数に影響します。このチェーンは、チェーンを検証して上に移動させるエンドエンティティー証明書で始まります。 拡張がエンドエンティティー証明書に設定されている場合、このパラメーターは機能しません。 許容値は **0** または **n** です。値は、CA 署名証明書の Basic Constraints 拡張で指定されたパスの長さよりも大きくする必要があります。 **0** は、発行されている従属 CA 証明書の下に従属 CA 証明書を許可しないことを指定します。パスをたどることができるのは、エンドエンティティー証明書のみです。 **n** は、ゼロよりも大きい整数でなければなりません。これは、使用されている従属 CA 証明書の下で許可される従属 CA 証明書の最大数です。 フィールドが空白の場合、パスの長さはデフォルトで、発行者の証明書の Basic Constraint 拡張機能で設定されたパスの長さによって決定されます。発行者のパスの長さが無制限の場合は、下位 CA 証明書のパスの長さも無制限です。発行者のパス長がゼロより大きい整数の場合、下位 CA 証明書のパス長は、発行者のパス長より 1 小さい値に設定されます。たとえば、発行者のパス長が 4 の場合、下位 CA 証明書のパス長は 3 に設定されます。

B.2.2. CA Validity 制約

CA 有効性制約は、証明書テンプレートの有効期間が CA の有効期間内にあるかどうかをチェックします。証明書の有効期間が CA 証明書の有効期間外である場合は、制約が拒否されます。

B.2.3. 拡張された鍵使用拡張制約

Extended Key Usage 拡張制約は、証明書の Extended Key Usage 拡張機能がこの制約で設定された基準を満たしているかどうかを確認します。

表B.26 拡張された主な使用拡張制約設定パラメーター

パラメーター	説明
exKeyUsageCritical	**true** に設定すると、エクステンションは重要であるとマークできます。**false** に設定すると、拡張は重要でないとしてマークできます。
exKeyUsageOIDs	キーの使用目的を特定する許容可能な OID を指定します。複数の OID をコンマ区切りの一覧に追加できます。

B.2.4. 拡張制約

この制約は、一般的な拡張制約を実装します。拡張機能が存在するかどうかを確認します。

表B.27 拡張制約

パラメーター	説明
extCritical	エクステンションは critical または noncritical のマークを付けるかどうかを指定します。**true** を選択して拡張機能を重要 (Critical) とマークします。**false** を選択して、非クリティカルにマークします。**-** を選択して、優先なしを強制します。
extOID	制約を渡すために証明書に存在する必要がある拡張の OID。

B.2.5. 主要な制約

この制約は、RSA キーのキーのサイズと、EC キーの楕円曲線の名前を確認します。RSA キーと一緒に使用すると、***KeyParameters*** パラメーターには、有効なキーサイズのコンマ区切りリストが含まれ、EC キーの場合は ***KeyParameters*** パラメーターには、使用可能な ECC 曲線のコンマ区切りのリストが含まれています。

表B.28 キー制約の設定パラメーター

パラメーター	説明

パラメーター	説明
keyType	キーの種類を指定します。これはデフォルトで **-** に設定されており、RSA キーシステムを使用します。rsa と ec を選択できます。キータイプが指定され、システムで識別されていない場合、制約は拒否されます。
KeyParameters	特定のキーパラメーターを定義します。キーに設定されるパラメーターは、***keyType*** パラメーターの値によって異なります (つまり、キータイプによって異なります)。 ● RSA 鍵では、***KeyParameters*** パラメーターに有効な鍵サイズのコンマ区切りリストが含まれます。 ● ECC キーでは、***KeyParameters*** パラメーターに、利用可能な ECC 曲線のコンマ区切りリストが含まれます。

B.2.6. 主な使用拡張機能の制約

Key Usage 拡張制約は、証明書要求のキー使用制約がこの制約で設定された基準を満たしているかどうかを確認します。

表B.29 主な使用拡張制約設定パラメーター

パラメーター	説明
keyUsageCritical	**true** を選択して、この拡張機能を重要としてマークします。**false** を選択して、非クリティカルにマークします。設定しない場合は **-** を選択します。
keyUsageDigitalSignature	SSL クライアント証明書と S/MIME 署名証明書を許可するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

パラメーター	説明
kleyUsageNonRepudiation	S/MIME 署名証明書を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。 **警告** このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。
keyEncipherment	SSL サーバー証明書と S/MIME 暗号化証明書の拡張子を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageDataEncipherment	キーマテリアルの代わりに、サブジェクトの公開キーを使用してユーザーデータを暗号化するときに、拡張子を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageKeyAgreement	サブジェクトの公開鍵がキー合意に使用されるたびに拡張を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

パラメーター	説明
keyUsageCertsign	この機能がすべての CA 署名証明書に適用されるかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageCRLSign	CRL に署名するのに使用する CA 署名証明書の拡張を設定するかどうかを指定します。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageEncipherOnly	公開鍵を使用してデータの暗号化のみに使用する場合に、拡張機能を設定するかどうかを指定します。このビットが設定されている場合は、**keyUsageKeyAgreement** も設定する必要があります。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。
keyUsageDecipherOnly	公開鍵をデータの解読にのみ使用する場合に、拡張子を設定するかどうかを指定します。このビットが設定されている場合は、**keyUsageKeyAgreement** も設定する必要があります。この値を set としてマークするには **true** を選択します。選択されないようにするには **false** を選択します。このパラメーターに制約がないことを示すには、ハイフン (**-**) を選択します。

B.2.7. Netscape Certificate Type 拡張機能の制約

警告

この制約は廃止されました。Netscape Certificate Type 拡張制約を使用する代わりに、Key Usage 拡張または Extended Key Usage 拡張を使用します。

Netscape Certificate Type 拡張制約は、証明書要求の Netscape Certificate Type 拡張がこの制約で設定された基準を満たしているかどうかをチェックします。

B.2.8. No Constraint

この制約は、制約が実装されていません。デフォルトとともに選択すると、そのデフォルトには制約は含まれません。

B.2.9. Renewal Grace Period 制約

Renewal Grace Period Constraint は、ユーザーが有効期限に基づいて証明書を更新できる時期に関するルールを設定します。たとえば、ユーザーは、有効期限が切れる前の特定の時間まで、または有効期限後の特定の時間を過ぎると、証明書を更新できません。

この制約を使用するときに覚えておくべき重要なことの 1 つは、この制約が更新プロファイルではなく、**元の登録プロファイル** に設定されていることです。更新猶予期間のルールは元の証明書の一部であり、引き継がれ、その後の更新に適用されます。

この制約は、No Default 拡張機能でのみ使用できます。

表B.30 Renewal Grace Period 制約設定パラメーター

パラメーター	説明
renewal.graceAfter	証明書の期限が切れた **後**、更新用に提出できる期間を日数単位で設定します。証明書の有効期限が切れると、更新要求は拒否されます。値を指定しない場合、制限はありません。
renewal.graceBefore	証明書の期限が切れる **前**、更新用に提出できる期間を日数単位で設定します。証明書が有効期限にそれほど近くない場合、更新要求は拒否されます。値を指定しない場合、制限はありません。

B.2.10. アルゴリズム制約の署名

署名アルゴリズム制約は、証明書要求の署名アルゴリズムがこの制約で設定された基準を満たしているかどうかを確認します。

表B.31 アルゴリズム制約設定パラメーターの署名

パラメーター	説明

パラメーター	説明
signingAlgsAllowed	証明書の署名に指定できる署名アルゴリズムを設定します。アルゴリズムは以下のいずれか 1 つになります。 • MD2withRSA • MD5withRSA • SHA256withRSA • SHA512withRSA • SHA256withEC • SHA384withEC • SHA512withEC

B.2.11. Subject Name 制約

Subject Name 制約は、証明書要求のサブジェクト名が基準を満たしているかどうかを確認します。

表B.32 Subject Name 制約設定パラメーター

パラメーター	説明
Pattern	サブジェクト DN を構築する正規表現または他の文字列を指定します。

Subject Name 名と正規表現

Subject Name Constraint の正規表現は、正規表現を照合するための Java 機能によって照合されます。これらの正規表現の形式は、https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html に記載されています。これにより、アスタリスク (*) などのワイルドカードは、任意の数の文字とピリオド (.) が、任意のタイプの文字を検索します。

たとえば、サブジェクト名制約のパターンが **uid=.*** に設定されている場合、証明書プロファイルフレームワークは、証明書要求のサブジェクト名がパターンと一致するかどうかを確認します。**uid=user, o=Example, c=US** のようなサブジェクト名は、**uid=.*** というパターンに対応します。サブジェクト名 **cn=user, o=example,c=US** はパターンを満たしません。**uid=.*** は、サブジェクト名は、***uid*** 属性で始まる必要があることを意味します。ピリオドアスタリスク (.*) ワイルドカードを使用すると、***uid*** の後に任意のタイプと文字数文字を続けることができます。

.*ou=Engineering.* などの内部パターンが必要になる場合があります。これには、その前後で任意の種類の文字列を持つ **ou=Engineering** 属性が必要になります。これは **cn=jdoe,ou=internal,ou=west coast,ou=engineering,o="Example Corp",st=NC** と同様に **uid=bjensen,ou=engineering,dc=example,dc=com** と一致します。

最後に、オプションの間にパイプ記号 (|) を設定することで、ある文字列または別の文字列のリクエストを許可することもできます。たとえば、**ou=engineering,ou=people** または **ou=engineering,o="Example Corp"** のいずれかが含まれるサブジェクト名を許可するには、パターンは **.*ou=engineering,ou=people.* | .*ou=engineering,o="Example Corp".*** のいずれかになります。

注記

ピリオド (.) などの特殊文字を使用するパターンを作成する場合は、バックスラッシュ (\) で文字をエスケープします。たとえば、文字列 **o="Example Inc."** を検索するには、パターンを **o="Example Inc\."** に設定します。

証明書の要求における Subject Name および UID または CN

サブジェクト DN を構築するために使用されるパターンは、証明書を要求する人の CN または UID に基づくこともできます。Subject Name Constraint は、証明書要求の DN で認識する CN (または UID) のパターンを設定し、Subject Name Default は、その CN に、事前定義されたディレクトリーツリーを使用して証明書のサブジェクト DN を作成します。

たとえば、証明書要求の CN を使用するには、次を実行します。

```
policyset.serverCertSet.1.constraint.class_id=subjectNameConstraintImpl
policyset.serverCertSet.1.constraint.name=Subject Name Constraint
policyset.serverCertSet.1.constraint.params.pattern=CN=[^,]+,.+
policyset.serverCertSet.1.constraint.params.accept=true
policyset.serverCertSet.1.default.class_id=subjectNameDefaultImpl
policyset.serverCertSet.1.default.name=Subject Name Default
policyset.serverCertSet.1.default.params.name=CN=$request.req_subject_name.cn$,DC=example,
DC=com
```

B.2.12. Unique Key 制約

この制約は、公開鍵が一意であることを確認します。

表B.33 Unique Key の制約パラメーター

パラメーター	説明
allowSameKeyRenewal	公開鍵は一意でなく、**かつ** サブジェクト DN が既存の証明書と一致し、このパラメーターが **true** に設定されている場合、リクエストは更新と見なされ、受け入れられます。ただし、公開鍵が重複していて、既存の Subject DN と一致しない場合、要求は拒否されます。 このパラメーターが **false** に設定されていると、重複した公開鍵リクエストは拒否されます。

B.2.13. Unique Subject Name 制約

Unique Subject Name 制約は、サーバーが同じサブジェクト名で複数の証明書を発行することを制限します。証明書要求が送信されると、サーバーは自動的にニックネームを他の発行された証明書のニックネームと照合します。この制約は、エンドエンティティーのページからの証明書の登録と更新に適用できます。

1 つの証明書の有効期限が切れているか取り消されていない限り (保留されていない場合)、証明書に同じサブジェクト名を付けることはできません。したがって、アクティブな証明書はサブジェクト名を共有できません。ただし、例外が 1 つあります。証明書のキー使用ビットが異なる場合は、使用法が異なるため、同じサブジェクト名を共有できます。

表B.34 Unique Subject Name 制約設定パラメーター

パラメーター	説明
enableKeyUsageExtensionChecking	キーの使用設定が異なる限り、証明書が同じサブジェクト名を持つことを許可するオプションの設定。これは **true** または **false** です。デフォルトは **true** で、重複したサブジェクト名を許可します。

B.2.14. Validity 制約

Validity 制約は、証明書要求の有効期間が基準を満たしているかどうかを確認します。

提供されるパラメーターは、適切な値でなければなりません。たとえば、すでに経過した時間を提供する ***notBefore*** パラメーターは受け入れず、 ***notBefore*** 時間よりも早い時間を提供する ***notAfter*** パラメーターは受け入れません。

表B.35 Validity 制約の設定パラメーター

パラメーター	説明
range	有効期間の範囲。日数を設定する整数です。***notBefore*** 時間と ***notAfter*** 時間の差 (日数) は範囲値よりも短くする必要があります。そうでない場合、この制約は拒否されます。
notBeforeCheck	範囲が猶予期間内ではないことを確認します。ブール値パラメーター ***NotBeforeCheck*** が true に設定されている場合、システムは、***notBefore***時間が、現在の時間に ***notBeforeGracePeriod*** 値を加えた値より大きくないことを確認します。***notBeforeTime*** が、現在の時間と ***notBeforeGracePeriod*** 値の間になければ、この制約は拒否されます。
notBeforeGracePeriod	***notBefore*** 時間後の猶予期間 (秒単位)。***notBeforeTime*** が、現在の時間と ***notBeforeGracePeriod*** 値の間になければ、この制約は拒否されます。この制約は、***notBeforeCheck*** パラメーターが true に設定されている場合にのみ確認されます。
notAfterCheck	指定された時間が期限切れの期間後にないかどうかを確認します。ブール値パラメーターが ***notAfterCheck*** を true に設定されている場合、システムは ***notAfter*** 時間は現在の時間より大きくありません。現在の時間がこの ***notAfter*** 時間を超えると、この制約は拒否されます。

B.3. 標準仕様の X.509 V3 証明書拡張機能リファレンス

X.509 v3 証明書には、証明書に任意の数の追加フィールドを追加できる拡張フィールドが含まれています。証明書拡張は、代替サブジェクト名や使用制限などの情報を証明書に追加する方法を提供しま

す。Red Hat Directory Server や Red Hat Certificate System などの古い Netscape サーバーは、PKIX パート 1 標準が定義される前に開発されたため、Netscape 固有の拡張機能が必要です。

以下は、X.509 v3 拡張機能が含まれる証明書のセクションの例です。Certificate System は、以下に示すように、読み取り可能な pretty-print 形式で証明書を表示できます。この例のように、証明書拡張は順番に表示され、証明書ごとに特定の拡張子のインスタンスが 1 つだけ表示される場合があります。たとえば、証明書に含まれるサブジェクトキー識別子の拡張子は 1 つだけです。これらの拡張機能をサポートする証明書には、バージョン **0x2** があります (これは、バージョン 3 に対応します)。

例B.4 Pretty-Print 証明書拡張の例

```
Data:
  Version:  v3
  Serial Number: 0x1
  Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
  Issuer: CN=Certificate Manager,OU=netscape,O=ExampleCorp,L=MV,ST=CA,C=US
  Validity:
    Not Before: Friday, February 21, 2005 12:00:00 AM PST America/Los_Angeles
    Not  After: Monday, February 21, 2007 12:00:00 AM PST America/Los_Angeles
  Subject: CN=Certificate Manager,OU=netscape,O=ExampleCorp,L=MV,ST=CA,C=US
  Subject Public Key Info:
    Algorithm: RSA - 1.2.840.113549.1.1.1
    Public Key:
      Exponent: 65537
      Public Key Modulus: (2048 bits) :
        E4:71:2A:CE:E4:24:DC:C4:AB:DF:A3:2E:80:42:0B:D9:
        CF:90:BE:88:4A:5C:C5:B3:73:BF:49:4D:77:31:8A:88:
        15:A7:56:5F:E4:93:68:83:00:BB:4F:C0:47:03:67:F1:
        30:79:43:08:1C:28:A8:97:70:40:CA:64:FA:9E:42:DF:
        35:3D:0E:75:C6:B9:F2:47:0B:D5:CE:24:DD:0A:F7:84:
        4E:FA:16:29:3B:91:D3:EE:24:E9:AF:F6:A1:49:E1:96:
        70:DE:6F:B2:BE:3A:07:1A:0B:FD:FE:2F:75:FD:F9:FC:
        63:69:36:B6:5B:09:C6:84:92:17:9C:3E:64:C3:C4:C9
  Extensions:
    Identifier: Netscape Certificate Type - 2.16.840.1.113730.1.1
      Critical: no
      Certificate Usage:
        SSL CA
        Secure Email CA
        ObjectSigning CA
    Identifier: Basic Constraints - 2.5.29.19
      Critical: yes
      Is CA: yes
      Path Length Constraint: UNLIMITED
    Identifier: Subject Key Identifier - 2.5.29.14
      Critical: no
      Key Identifier:
        3B:46:83:85:27:BC:F5:9D:8E:63:E3:BE:79:EF:AF:79:
        9C:37:85:84
    Identifier: Authority Key Identifier - 2.5.29.35
      Critical: no
      Key Identifier:
        3B:46:83:85:27:BC:F5:9D:8E:63:E3:BE:79:EF:AF:79:
        9C:37:85:84
    Identifier: Key Usage: - 2.5.29.15
      Critical: yes
```

```
        Key Usage:
          Digital Signature
          Key CertSign
          Crl Sign
    Signature:
      Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
      Signature:
        AA:96:65:3D:10:FA:C7:0B:74:38:2D:93:54:32:C0:5B:
        2F:18:93:E9:7C:32:E6:A4:4F:4E:38:93:61:83:3A:6A:
        A2:11:91:C2:D2:A3:48:07:6C:07:54:A8:B8:42:0E:B4:
        E4:AE:42:B4:B5:36:24:46:4F:83:61:64:13:69:03:DF:
        41:88:0B:CB:39:57:8C:6B:9F:52:7E:26:F9:24:5E:E7:
        BC:FB:FD:93:13:AF:24:3A:8F:DB:E3:DC:C9:F9:1F:67:
        A8:BD:0B:95:84:9D:EB:FC:02:95:A0:49:2C:05:D4:B0:
        35:EA:A6:80:30:20:FF:B1:85:C8:4B:74:D9:DC:BB:50
```

オブジェクト識別子 (OID) は、証明書の拡張子や会社の証明書運用明細書など、一意のオブジェクトを識別する数字の文字列です。Certificate System には、サーバーが発行する証明書に X.509 証明書拡張を追加できるようにする拡張固有のプロファイルプラグインモジュールのセットが付属しています。一部の拡張機能には、OID を指定するためのフィールドが含まれています。

PKIX 標準では、証明書で使用される拡張子やステートメントなどのすべてのオブジェクトを OID の形式で含めることを推奨しています。これにより、共有ネットワーク上の組織間の相互運用性が促進されます。共有ネットワークで使用される証明書が発行される場合は、OID 接頭辞を適切な登録機関に登録してください。

OID は、国際標準組織 (ISO) 登録機関によって制御されます。場合によっては、この権限は ISO から地域の登録機関に委任されます。米国では、American National Standards Institute (ANSI) がこの登録を管理します。

別の組織に登録されている OID を使用したり、OID の登録に失敗したりすると、状況によっては法的な結果が生じる可能性があります。登録には手数料がかかる場合があります。詳細は、適切な登録認証局にお問い合わせください。

カスタムオブジェクトの OID を定義または割り当てるには、会社の *アーク*、つまり民間企業の OID を知ってください。会社にアーカイブがない場合は、ファイルを取得する必要があります。http://www.alvestrand.no/objectid/ には、OID の登録および使用に関する詳細情報が記載されています。

たとえば、**Netscape Certificate Comment** という名前の拡張の Netscape 定義の OID は 2.16.840.1.113730.1.13 です。この拡張機能に割り当てられた OID は階層的であり、以前の Netscape 社のアーク **2.16.840.1** が含まれています。OID 定義エントリーは http://www.alvestrand.no/objectid/2.16.840.1.113730.1.13.html です。

OID 拡張が証明書に存在し、クリティカルとマークされている場合、証明書を検証するアプリケーションは、任意の修飾子を含めて拡張を解釈できる必要があります。そうでない場合は、証明書を拒否する必要があります。すべてのアプリケーションが OID の形式で埋め込まれた会社のカスタム拡張機能を解釈できる可能性は低いため、PKIX 標準では、拡張機能を常に非クリティカルではないとマークすることを推奨しています。

このセクションでは、インターネット X.509 バージョン 3 標準の一部として定義されている拡張タイプを要約し、PKIX ワーキンググループが推奨するタイプを紹介します。

この参照では、各証明書に関する重要な情報をまとめています。完全な詳細については、ITU から入手可能な X.509 v3 の規格、および RFC 3280 で入手可能な『Internet X.509 Public Key Infrastructure -

Certificate and CRL Profile (RFC 3280)』 の両方を参照してください。拡張機能の説明は、拡張機能について説明している標準ドラフトの RFC とセクション番号を参照しています。各拡張機能のオブジェクト識別子 (OID) も提供しています。

証明書の各拡張は、クリティカルまたは非クリティカルとして指定できます。Web ブラウザーなどの証明書を使用するシステムは、認識できない重要な拡張子に遭遇した場合、証明書を拒否する必要があります。ただし、重要でない拡張子は、認識されない場合は無視できます。

B.3.1. authorityInfoAccess

Authority Information Access 拡張機能は、証明書の発行者に関する情報にアクセスする方法と場所を示します。エクステンションには **accessMethod** および **accessLocation** フィールドが含まれます。**accessMethod** は、**accessLocation** という名前の発行者に関する情報のタイプおよび形式を OID に指定します。

PKIX Part 1 は、**accessMethod** (**id-ad-calssuers**) を 1 つ定義して、この拡張機能を使用して、証明書の発行者よりも CA チェーンの上位にある証明書を発行した CA のリストを取得します。**accessLocation** フィールドには、通常、リストの取得に使用される場所とプロトコル (LDAP、HTTP、または FTP) を示す URL が含まれます。

RFC 2560 で入手可能な Online Certificate Status Protocol (RFC 2560) は、OCSP を使用して証明書を検証するために *accessMethod* (**id-ad-ocsp**) を定義します。次に、**accessLocation** フィールドには、証明書を検証できる OCSP レスポンダーへのアクセスに使用される場所とプロトコルを示す URL が含まれます。

OID

1.3.6.1.5.5.7.1.1

Criticality

この拡張はクリティカルでない必要があります。

B.3.2. authorityKeyIdentifier

Authority Key Identifier 拡張機能は、証明書の署名に使用される秘密鍵に対応する公開鍵を識別します。この拡張機能は、CA 証明書が更新される場合など、発行者が複数の署名キーを持っている場合に役立ちます。

拡張機能は、次のいずれかまたは両方で設定されます。

- **keyIdentifier** フィールドに設定される明示的なキー識別子
- **authorityCertSerialNumber** フィールドで設定、シリアル番号、**authorityCertSerialNumber** フィールドで設定、証明書を識別。

keyIdentifier フィールドが存在する場合は、一致する **subjectKeyIdentifier** 拡張子を持つ証明書を選択するために使用されます。**AuthorityCertIssuer** および **authorityCertSerialNumber** フィールドが存在する場合、それらは **issuer** および **serialNumber** によって正しい証明書を識別するために使用されます。

この拡張子が存在しない場合は、発行者名のみが発行者証明書の識別に使用されます。

PKIX Part 1 では、自己署名ルート CA 証明書を除くすべての証明書にこの拡張機能が必要です。キー識別子が確立されていない場合、PKIX は **authorityCertIssuer** および **authorityCertSerialNumber** フィールドを指定することが推奨されます。これらのフィールドにより、発行元証明書の

authortiyCertIssuer および **authorityCertSerialNumber** 拡張子で、発行者の証明書内の **SubjectName** フィールドおよび **CertificateSerialNumber** フィールドを照合して完全な証明書チェーンを作成できます。

OID

2.5.29.35

Criticality

この拡張は常に非クリティカルではなく、常に評価されます。

B.3.3. basicConstraints

この拡張機能は、証明書チェーンの検証プロセス中に、CA 証明書を識別し、証明書チェーンパスの長さの制約を適用するために使用されます。**cA** コンポーネントは、すべての CA 証明書に対して **true** に設定する必要があります。PKIX は、この拡張がエンドエンティティーの証明書に表示されないことを推奨します。

pathLenConstraint コンポーネントが存在する場合、その値は、エンドエンティティー証明書から始まり、チェーンを上に移動して、これまでに処理された CA 証明書の数よりも大きくなければなりません。**pathLenConstraint** を省略し、拡張機能が存在する場合は、チェーン内のすべての上位レベルの CA 証明書にこのコンポーネントを含めることはできません。

OID

2.5.29.19

Criticality

PKIX Part 1 では、この拡張が critical とマークされている必要があります。この拡張機能は、その重要度に関係なく評価されます。

B.3.4. certificatePoliciesExt

Certificate Policies 拡張機能は、1 つ以上のポリシーを定義します。各ポリシーは、OID と任意の修飾子で設定されます。拡張機能には、発行者の Certificate Practice Statement への URI を含めることも、ユーザー通知などの発行者情報をテキスト形式で埋め込むこともできます。この情報は、証明書が有効なアプリケーションで使用できます。

この拡張機能が存在する場合、PKIX Part 1 では、ポリシーを OID のみで識別するか、必要に応じて特定の推奨修飾子のみで識別することを推奨しています。

OID

2.5.29.32

Criticality

この拡張機能は、重要な場合と重要でない場合があります。

B.3.5. CRLDistributionPoints

この拡張は、CRL 情報の取得方法を定義します。システムが CRL 発行ポイントを使用するように設定されている場合は、これを使用する必要があります。

拡張機能に、タイプが URI に設定されている **DistributionPointName** が含まれている場合は、指定された失効理由のために現在の CRL へのポインターであると見なされ、**cRLIssuer** という名前で発行さ

れます。URI の想定される値は、Subject Alternative Name 拡張に定義される値です。**distributionPoints** の理由を省略する場合は、すべての理由で CRL の取り消しを含める必要があります。**distributionPoint** が **cRLIssuer** を省略する場合、証明書を発行した CA によって CRL を発行する必要があります。

PKIX は、CA およびアプリケーションでこの拡張をサポートすることを推奨します。

OID

2.5.29.31

Criticality

PKIX では、この拡張機能を非クリティカルとマークし、すべての証明書でサポートすることが推奨されます。

B.3.6. extKeyUsage

Extended Key Usage 拡張機能は、認証された公開キーを使用できる目的を示します。これらの目的は、Key Usage 拡張機能に示されている基本的な目的に加えてまたはその代わりになる場合があります。

Extended Key Usage 拡張機能には、レスポンダーによって検証された証明書に署名した CA 署名キーが OCS P 署名キーでもない限り、OCSP レスポンダーの証明書内の **OCSP Signing** が含まれる必要があります。OCSP レスポンダーの証明書は、レスポンダーが検証する証明書に署名する CA によって直接発行される必要があります。

Key Usage、Extended Key Usage、および Basic Constraints 拡張機能は連携して機能し、証明書の使用目的を定義します。アプリケーションはこれらの拡張機能を使用して、不適切なコンテキストでの証明書の使用を禁止できます。

表B.36「PKIX Extended Key Usage Extension の使用」 は、この拡張機能に対して PKIX によって定義された用途を一覧表示します。表B.37「Private Extended Key Usage Extension の使用」 は、Netscape によって非公開で定義されたものを使用します。

OID

2.5.29.37

Criticality

この拡張機能がクリティカルとマークされている場合、証明書は指定された目的の 1 つにのみ使用する必要があります。クリティカルとマークされていない場合は、キーを識別するために使用できるアドバイザリーフィールドとして扱われますが、証明書の使用は指定された目的に制限されません。

表B.36 PKIX Extended Key Usage Extension の使用

使用	OID
サーバー認証	1.3.6.1.5.5.7.3.1
クライアント認証	1.3.6.1.5.5.7.3.2
コード署名	1.3.6.1.5.5.7.3.3
Email	1.3.6.1.5.5.7.3.4

使用	OID
タイムスタンプ	1.3.6.1.5.5.7.3.8
OCSP 署名	1.3.6.1.5.5.7.3.9[a]
[a] OCSP 署名は PKIX Part 1 では定義されていませんが、RFC 2560 『X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP』 では定義されていません。	

表B.37 Private Extended Key Usage Extension の使用

使用	OID
証明書トラストリスト署名	1.3.6.1.4.1.311.10.3.1
Microsoft Server Gated Crypto (SGC)	1.3.6.1.4.1.311.10.3.3
Microsoft 暗号化ファイルシステム	1.3.6.1.4.1.311.10.3.4
Netscape SGC	2.16.840.1.113730.4.1

B.3.7. issuerAltName 拡張

Issuer Alternative Name 拡張は、インターネットスタイルのアイデンティティーを証明書発行者に関連付けるために使用されます。名前には、Subject Alternative Name 拡張機能に定義したフォームを使用する必要があります。

OID

2.5.29.18

Criticality

PKIX Part 1 は、この拡張を非クリティカルとしてマークすることを推奨します。

B.3.8. keyUsage

Key Usage 拡張機能は、証明書に含まれるキーの目的を定義します。Key Usage、Extended Key Usage、および Basic Constraints 拡張機能は連携して機能し、証明書を使用できる目的を指定します。

この拡張機能が全く含まれる場合は、以下のようにビットを設定します。

- SSL クライアント証明書、S/MIME 署名証明書、およびオブジェクト署名証明書用の **digitalSignature** (**0**)。
- 一部の S/MIME 署名証明書およびオブジェクト署名証明書用の **nonRepudiation** (**1**)

警告

このビットの使用は議論の的になっています。証明書に設定する前に、その使用による法的影響を慎重に検討してください。

- これは、SSL サーバー証明書および S/MIME 暗号化証明書の **keyEncipherment** (**2**) に設定されます。
- **dataEncipherment** (**3**) サブジェクトの公開鍵を使用して、鍵情報ではなくユーザーデータを暗号化するとき。
- **keyAgreement** (**4**) サブジェクトの公開鍵がキー合意に使用される場合。
- すべての CA 署名証明書用の **keyCertSign** (**5**)。
- CRL の署名に使用される CA 署名証明書の **cRLSign** (**6**)。
- **encipherOnly** (**7**) 公開鍵がデータの暗号化にのみ使用される場合。このビットが設定されている場合 、 **keyAgreement** も設定する必要があります。
- **decipherOnly** (**8**) 公開鍵がデータの暗号化にのみ使用される場合。このビットが設定されている場合 、 **keyAgreement** も設定する必要があります。

表B.38「証明書の使用とそれに対応する鍵の使用方法」 は、通常の証明書使用ガイドラインをまとめています。

keyUsage 拡張機能が存在してクリティカルとマークされる場合は、証明書とキーの使用を強制するために使用されます。拡張機能は、キーの使用を制限するために使用されます。拡張機能が存在しないか非クリティカルな場合は、すべてのタイプの使用が許可されます。

keyUsage 拡張機能が存在する場合、クリティカルかどうかに関係なく、特定の操作に対して複数の証明書から選択するために使用されます。たとえば、操作用に個別の証明書とキーペアを持っているユーザーの個別の署名証明書と暗号化証明書を区別するために使用されます。

OID

2.5.29.15

Criticality

この拡張機能は、重要な場合と重要でない場合があります。PKIX Part 1 を使用する場合は、クリティカルなマークを付けることが推奨されます。

表B.38 証明書の使用とそれに対応する鍵の使用方法

証明書の目的	必要な鍵使用量のビット
CA 署名	• keyCertSign • cRLSign

証明書の目的	必要な鍵使用量のビット
SSL クライアント	digitalSignature
SSL Server	keyEncipherment
S/MIME 署名	digitalSignature
S/MIME 暗号化	keyEncipherment
証明書の署名	keyCertSign
オブジェクトの署名	digitalSignature

B.3.9. nameConstraints

この拡張機能は、CA 証明書でのみ使用でき、証明書パス内の後続の証明書のすべてのサブジェクト名を配置する必要がある名前空間を定義します。

OID

2.5.29.30

Criticality

PKIX Part 1 では、この拡張が critical とマークされている必要があります。

B.3.10. OCSPNocheck

拡張は OCSP 署名証明書に含まれることを目的としています。拡張機能は、(応答は OCSP レスポンダーによって再度署名され、クライアントは署名証明書の有効性ステータスを再度要求するため) OCSP レスポンダーに照会せずに署名証明書を信頼できることを OCSP クライアントに通知します。このエクステンションは null 値であり、その意味は存在するか、または存在しない場合により決定されます。

証明書にこの拡張機能が存在すると、OCSP クライアントはその証明書で署名された応答を信頼するため、この拡張機能の使用は慎重に管理する必要があります。OCSP 署名キーが危険にさらされると、証明書の有効期間中、PKI で証明書を検証するプロセス全体が危険にさらされます。したがって、**OCSPNocheck** を使用する証明書は短期間発行され、頻繁に更新される必要があります。

OID

1.3.6.1.5.5.7.48.4

Criticality

この拡張はクリティカルではありません。

B.3.11. policyConstraints

この拡張 (CA 証明書のみ) は、パスの検証を 2 つの方法で制限します。ポリシーマッピングを禁止したり、パス内の各証明書に受け入れ可能なポリシー識別子が含まれていることを要求したりするために使用できます。

PKIX は、(存在する場合)、この拡張は null シーケンスで設定されるべきではありません。利用可能な 2 つのフィールドが少なくとも 1 つ存在している必要があります。

OID

2.5.29.36

Criticality

この拡張機能は、重要な場合と重要でない場合があります。

B.3.12. policyMappings

Policy Mappings の拡張は、CA 証明書でのみ使用されます。これは、ある CA の対応するポリシーが別の CA のポリシーと同等であることを示すために使用される OID の 1 つ以上のペアをリストします。これは、ペア間の証明書のコンテキストで役に立つ場合があります。

この拡張機能は CA およびアプリケーションでサポートされる可能性があります。

OID

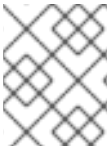
2.5.29.33

Criticality

この拡張はクリティカルでない必要があります。

B.3.13. privateKeyUsagePeriod

Private Key Usage Period の拡張機能により、証明書発行者は、証明書自体に秘密鍵に異なる有効期間を指定できます。この拡張は、デジタル署名鍵の使用を目的としています。



注記

PKIX Part 1 はこの拡張機能の使用に対して推奨されます。PKIX Part 1 に準拠する CA は、この拡張で証明書を生成 **することはできません。**

OID

2.5.29.16

B.3.14. subjectAltName

Subject Alternative Name 拡張には、CA によって認証された公開鍵にバインドされた ID の 1 つ以上の代替 (X.500 以外) 名が含まれます。証明書サブジェクト名に加えて、またはその代わりとして使用できます。定義された名前の形式には、インターネット電子メールアドレス (RFC-822 で定義された SMTP)、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、および URI (Uniform Resource Identifier) が含まれます。

PKIX では、サブジェクトフィールドで使用される X.500 識別名 (DN) 以外の名前形式で識別されるエンティティーに対してこの拡張機能が必要です。PKIX Part 1 では、この拡張機能とサブジェクトフィールドの関係に関する追加のルールを説明します。

電子メールアドレスは、Subject Alternative Name 拡張機能、証明書のサブジェクト名フィールド、またはその両方で指定できます。メールアドレスが件名の一部である場合は、PKCS #9 で定義された属性 **EmailAddress** の形式である必要があります。S/MIME をサポートするソフトウェアは、サブジェク

ト代替名拡張子またはサブジェクト名フィールドのいずれかから電子メールアドレスを読み取ることができる必要があります。

OID

2.5.29.17

Criticality

証明書の subject フィールドが空の場合は、この拡張機能にクリティカルのマークが付けられている必要があります。

B.3.15. subjectDirectoryAttributes

Subject Directory Attributes 機能拡張は、証明書の件名に必要なディレクトリー属性の値をすべて伝えます。提案されている PKIX 標準の必須部分としては推奨されていませんが、ローカル環境で使用できます。

OID

2.5.29.9

Criticality

PKIX Part 1 では、この拡張が非クリティカルなものとしてマークされている必要があります。

B.3.16. subjectKeyIdentifier

Subject Key Identifier 拡張は、この証明書で認定された公開鍵を識別します。この拡張機能は、特定のサブジェクト名に複数の公開鍵が使用可能な場合に、公開鍵を区別する方法を提供します。

この拡張機能の値は、PKIX で推奨されているとおり、証明書の DER エンコードされた **subjectPublicKey** の SHA-1 ハッシュを実行して計算する必要があります、Subject Key Identifier 拡張機能は、CA 証明書の Authority Key Identifier 拡張機能と組み合わせて使用されます。CA 証明書に Subject Key Identifier 拡張がある場合、検証される証明書の Authority Key Identifier のキー識別子は、CA の Subject Key Identifier 拡張のキー識別子と一致する必要があります。この場合、検証者がキー識別子を再計算する必要はありません。

PKIX Part 1 では、すべての CA 証明書にこの拡張機能が必要であり、他のすべての証明書にこの拡張機能を推奨しています。

OID

2.5.29.14

Criticality

この拡張機能は常にクリティカルではありません。

B.4. CRL 拡張機能

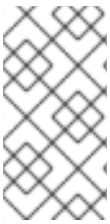
B.4.1. CRL 拡張について

最初の発行以来、CRL 形式の X.509 標準が修正され、CRL 内に追加情報が含まれるようになりました。この情報は CRL 拡張機能により追加されます。

ANSI X9 および ISO/IEC/ITU for X.509 CRLs [X.509] [X9.55] で定義されている拡張機能により、追加

の属性を CRL に関連付けることができます。RFC5280 で入手可能な『Internet X.509 Public Key Infrastructure Certificate and CRL Profile』 は、CRL で使用される拡張機能のセットを推奨しています。これらの拡張機能は *標準 CRL 拡張機能* と呼ばれます。

この規格では、カスタム拡張機能を作成して CRL に含めることもできます。これらの拡張機能は、 *プライベート* 拡張機能、 *プロプライエタリー* 拡張機能、または *カスタム* CRL 拡張機能と呼ばれ、組織またはビジネスに固有の情報を伝達します。アプリケーションは、プライベートの重要な拡張機能を含む CRL を検証できない場合があるため、一般的なコンテキストでカスタム拡張機能を使用することはお勧めしません。



注記

Abstract Syntax Notation One (ASN.1) および Distinguished Encoding Rules (DER) 標準は、CCITT 勧告 X.208 および X.209 で指定されています。ASN.1 および DER の概要については、RSA Laboratories の Web サイト (http://www.rsa.com) で入手できる『A Layman's Guide to a Subset of ASN.1, BER, and DER』を参照してください。

B.4.1.1. CRL 拡張の構造

CRL 拡張機能は、以下の部分で設定されます。

- 拡張のオブジェクト識別子 (OID)。この識別子は、拡張子を一意に識別します。また、値フィールドの ASN.1 タイプの値や、値が解釈される方法も決定します。拡張機能が CRL に表示されると、OID は拡張機能 ID フィールド (**extnID**) として示されます。また、対応する ASN.1 エンコード構造は、オクテット文字列の値 (**extnValue**) として示されます。例は 例 B.4「Pretty-Print 証明書拡張の例」 で示されます。

- **critical** というフラグまたはブール値フィールド

  このフィールドに割り当てられた **true** 値または **false** 値は、拡張機能が CRL にとってクリティカルか非クリティカルかを示します。

  - 拡張機能が重要であり、拡張機能の ID に基づいて拡張機能を理解しないアプリケーションに CRL が送信された場合は、アプリケーションが CRL を拒否する必要があります。

  - 拡張機能が重要ではなく、拡張機能の ID に基づいて拡張機能を理解しないアプリケーションに CRL が送信された場合、アプリケーションは拡張機能を無視して CRL を受け入れることができます。

- 拡張の値の DER エンコーディングを含む octet 文字列。

CRL を受信するアプリケーションは、拡張 ID を確認して、ID を認識できるかどうかを判断します。可能であれば、エクステンション ID を使用して、使用する値のタイプを判断します。

B.4.1.2. CRL および CRL エントリー拡張のサンプル

以下は、X.509 CRL バージョン 2 の拡張機能の例です。Certificate System は、以下に示すように、読み取り可能な pretty-print 形式で CRL を表示できます。例に示されているように、CRL 拡張は順番に表示され、特定の拡張の 1 つのインスタンスのみが CRL ごとに表示される場合があります。たとえば、CRL に含まれる Authority Key Identifier 拡張機能は 1 つだけです。ただし、CRL-entry 拡張は CRL の適切なエントリーに表示されます。

```
Certificate Revocation List:
    Data:
        Version:  v2
```

```
        Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
        Issuer: CN=Certificate Authority,O=Example Domain
        This Update: Wednesday, July 29, 2009 8:59:48 AM GMT-08:00
        Next Update: Friday, July 31, 2009 8:59:48 AM GMT-08:00
        Revoked Certificates: 1-3 of 3
            Serial Number: 0x11
            Revocation Date: Thursday, July 23, 2009 10:07:15 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Privilege_Withdrawn
            Serial Number: 0x1A
            Revocation Date: Wednesday, July 29, 2009 8:50:11 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Certificate_Hold
                Identifier: Invalidity Date - 2.5.29.24
                    Critical: no
                    Invalidity Date: Sun Jul 26 23:00:00 GMT-08:00 2009
            Serial Number: 0x19
            Revocation Date: Wednesday, July 29, 2009 8:50:49 AM GMT-08:00
            Extensions:
                Identifier: Revocation Reason - 2.5.29.21
                    Critical: no
                    Reason: Key_Compromise
                Identifier: Invalidity Date - 2.5.29.24
                    Critical: no
                    Invalidity Date: Fri Jul 24 23:00:00 GMT-08:00 2009
    Extensions:
        Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
            Critical: no
            Access Description:
                Method #0: ocsp
                Location #0: URIName: http://example.com:9180/ca/ocsp
        Identifier: Issuer Alternative Name - 2.5.29.18
            Critical: no
            Issuer Names:
                DNSName: example.com
        Identifier: Authority Key Identifier - 2.5.29.35
            Critical: no
            Key Identifier:
                50:52:0C:AA:22:AC:8A:71:E3:91:0C:C5:77:21:46:9C:
                0F:F8:30:60
        Identifier: Freshest CRL - 2.5.29.46
            Critical: no
            Number of Points: 1
            Point 0
                Distribution Point: [URIName: http://server.example.com:8443/ca/ee/ca/getCRL?
op=getDeltaCRL&crlIssuingPoint=MasterCRL]
        Identifier: CRL Number - 2.5.29.20
            Critical: no
            Number: 39
        Identifier: Issuing Distribution Point - 2.5.29.28
            Critical: yes
            Distribution Point:
```

```
            Full Name:
              URIName: http://example.com:9180/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL
        Only Contains User Certificates: no
        Only Contains CA Certificates: no
        Indirect CRL: no
  Signature:
    Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
    Signature:
      47:D2:CD:C9:E5:F5:9D:56:0A:97:31:F5:D5:F2:51:EB:
      1F:CF:FA:9E:63:D4:80:13:85:E5:D8:27:F0:69:67:B5:
      89:4F:59:5E:69:E4:39:93:61:F2:E3:83:51:0B:68:26:
      CD:99:C4:A2:6C:2B:06:43:35:36:38:07:34:E4:93:80:
      99:2F:79:FB:76:E8:3D:4C:15:5A:79:4E:E5:3F:7E:FC:
      D8:78:0D:1D:59:A0:4C:14:42:B7:22:92:89:38:3A:4C:
      4A:3A:06:DE:13:74:0E:E9:63:74:D0:2F:46:A1:03:37:
      92:F0:93:D9:AA:F8:13:C5:06:25:02:B0:FD:3B:41:E7:
      62:6F:67:A3:9F:F5:FA:03:41:DA:8D:FD:EA:2F:E3:2B:
      3E:F8:E9:CC:3B:9F:E4:ED:73:F2:9E:B9:54:14:C1:34:
      68:A7:33:8F:AF:38:85:82:40:A2:06:97:3C:B4:88:43:
      7B:AF:5D:87:C4:47:63:4A:11:65:E3:75:55:4D:98:97:
      C2:2E:62:08:A4:04:35:5A:FE:0A:5A:6E:F1:DE:8E:15:
      27:1E:0F:87:33:14:16:2E:57:F7:DC:77:BE:D2:75:AB:
      A9:7C:42:1F:84:6D:40:EC:E7:ED:84:F8:14:16:28:33:
      FD:11:CD:C5:FC:49:B7:7B:39:57:B3:E6:36:E5:CD:B6
```

デルタ CRL は、最後の CRL が公開されてからの変更のみを含む CRL のサブセットです。デルタ CRL インジケーター拡張を含む CRL はデルタ CRL です。

```
ertificate Revocation List:
  Data:
    Version:  v2
    Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
    Issuer: CN=Certificate Authority,O=SjcRedhat Domain
    This Update: Wednesday, July 29, 2009 9:02:28 AM GMT-08:00
    Next Update: Thursday, July 30, 2009 9:02:28 AM GMT-08:00
    Revoked Certificates:
      Serial Number: 0x1A
      Revocation Date: Wednesday, July 29, 2009 9:00:48 AM GMT-08:00
      Extensions:
        Identifier: Revocation Reason - 2.5.29.21
          Critical: no
          Reason: Remove_from_CRL
      Serial Number: 0x17
      Revocation Date: Wednesday, July 29, 2009 9:02:16 AM GMT-08:00
      Extensions:
        Identifier: Revocation Reason - 2.5.29.21
          Critical: no
          Reason: Certificate_Hold
        Identifier: Invalidity Date - 2.5.29.24
          Critical: no
          Invalidity Date: Mon Jul 27 23:00:00 GMT-08:00 2009
  Extensions:
    Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
      Critical: no
      Access Description:
```

```
            Method #0: ocsp
            Location #0: URIName: http://server.example.com:8443/ca/ocsp
        Identifier: Delta CRL Indicator - 2.5.29.27
          Critical: yes
          Base CRL Number: 39
        Identifier: Issuer Alternative Name - 2.5.29.18
          Critical: no
          Issuer Names:
            DNSName: a-f8.sjc.redhat.com
        Identifier: Authority Key Identifier - 2.5.29.35
          Critical: no
          Key Identifier:
            50:52:0C:AA:22:AC:8A:71:E3:91:0C:C5:77:21:46:9C:
            0F:F8:30:60
        Identifier: CRL Number - 2.5.29.20
          Critical: no
          Number: 41
        Identifier: Issuing Distribution Point - 2.5.29.28
          Critical: yes
          Distribution Point:
            Full Name:
              URIName: http://server.example.com:8443/ca/ee/ca/getCRL?
op=getCRL&crlIssuingPoint=MasterCRL
          Only Contains User Certificates: no
          Only Contains CA Certificates: no
          Indirect CRL: no
    Signature:
      Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
      Signature:
        68:28:DA:90:D5:39:CB:6D:BE:42:04:77:C9:E4:09:60:
        C1:97:A6:99:AB:A0:5B:A2:F3:8B:5E:4E:D6:05:70:B0:
        87:1F:D7:0E:4B:C6:B2:DE:8B:92:D8:7C:3B:36:1C:79:
        96:2A:64:E6:7A:25:1D:E7:40:62:48:7A:24:C9:9D:11:
        A6:7F:BB:6B:03:A0:9C:1D:BC:1C:EE:9A:4B:A6:48:2C:
        3B:5E:2B:B1:70:3C:C3:42:96:28:26:AB:82:18:F2:E9:
        F2:55:48:A8:7E:7F:FE:D4:3D:0B:EA:A2:2F:4E:E6:C3:
        C3:C1:6A:E5:C6:85:5B:42:B1:70:2A:C6:E1:D9:0C:AF:
        DA:01:22:FF:80:6E:2E:A7:E5:34:DC:AF:E6:C2:B5:B3:
        1B:FC:28:36:8A:91:4A:22:E7:03:A5:ED:4E:62:0C:D9:
        7F:81:BB:80:99:B8:61:2A:02:C6:9C:41:2E:01:82:21:
        80:82:69:52:BD:B2:AA:DB:0F:80:0A:7E:2A:F3:15:32:
        69:D2:40:0D:39:59:93:75:A2:ED:24:70:FB:EE:19:C0:
        BE:A2:14:36:D0:AC:E8:E2:EE:23:83:DD:BC:DF:38:1A:
        9E:37:AF:E3:50:D9:47:9D:22:7C:36:35:BF:13:2C:16:
        A2:79:CF:05:41:88:8E:B6:A2:4E:B3:48:6D:69:C6:38
```

B.4.2. 標準 X.509 v3 CRL 拡張機能リファレンス

証明書拡張に加えて、X.509 で提案されている標準では、CRL の拡張が定義されており、追加の属性をインターネット CRL に関連付ける方法が提供されています。これらは、CRL 自体の拡張と、CRL の個々の証明書エントリーの拡張の 2 種類のうちの 1 つです。

- 「CRL の拡張機能」
- 「CRL エントリー拡張」

B.4.2.1. CRL の拡張機能

以下の CRL の説明は、インターネット X.509 v3 公開鍵インフラストラクチャーが提案する標準の一部として定義されています。

- 「authorityInfoAccess」
- 「authorityKeyIdentifier」
- 「CRLNumber」
- 「deltaCRLIndicator」
- 「FreshestCRL」
- 「issuerAltName」
- 「issuingDistributionPoint」
- 「FreshestCRL」

B.4.2.1.1. authorityInfoAccess

Authority Information Access 拡張は、デルタ CRL 情報の取得方法を識別します。freshestCRL 拡張機能は完全な CRL に配置され、最新のデルタ CRL の場所を示します。

OID

1.3.6.1.5.5.7.1.1

Criticality

PKIX では、この拡張は重要ではありません。

パラメーター

表B.39 認証局情報アクセス設定パラメーター

パラメーター	説明
enable	ルールを有効または無効するかどうかを指定します。デフォルトでは、この拡張機能は無効になっています。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。
numberOfAccessDescriptions	0 から任意の正の整数までのアクセス記述の数を示します。デフォルトは 0 です。 このパラメーターを 0 以外の整数に設定する場合は、数値を設定し、OK をクリックしてウィンドウを閉じます。ルールの編集ウィンドウを再度開き、ポイントを設定するフィールドが存在します。

パラメーター	説明
accessMethod**n**	このパラメーターで許可される値は calssuers のみです。calssuers メソッドは、利用可能な情報に CRL の署名の検証に使用できる証明書がリストされている場合に使用されます。AIA 拡張が CRL に含まれる場合、他の方法は使用しないでください。
accessLocationType**n**	**n** アクセスの説明のアクセス場所を指定します。オプションは **DirectoryName** または **URI** です。
accessLocation**n**	**accessLocationType** が **DirectoryName** に設定されている場合、値は証明書のサブジェクト名と同様に、X.500 名の形式の文字列である必要があります。たとえば、 **CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 **accessLocationType** が **URI** に設定されている場合、名前は URI である必要があります。URI は絶対パス名で、ホストを指定する必要があります。例: **http://testCA.example.com/get/crls/here/**。

B.4.2.1.2. authorityKeyIdentifier

CRL の AuthorityKey Identifier 拡張機能は、CRL の署名に使用される秘密鍵に対応する公開鍵を識別します。詳細は、「authorityKeyIdentifier」 の証明書拡張を参照してください。

PKIX 標準では、CA の公開鍵は、たとえば鍵が更新されたときに変更される可能性があるため、または複数の同時鍵ペアまたは鍵切り替えのために CA が複数の署名鍵を持つ可能性があるため、CA が発行するすべての CRL にこの拡張機能を含める必要があることを推奨しています。このような場合、CA は複数のキーペアで終了します。証明書の署名を検証する場合、他のアプリケーションは、署名で使用されたキーを知る必要があります。

OID

2.5.29.35

パラメーター

表B.40 AuthorityKeyIdentifierExt 設定パラメーター

パラメーター	説明
enable	ルールを有効または無効するかどうかを指定します。デフォルトでは、この拡張機能は無効になっています。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。

B.4.2.1.3. CRLNumber

CRLNumber 拡張は、CA が発行する各 CRL の連続する番号を指定します。ユーザーは、特定の CRL が別の CRL を上書きするタイミングを簡単に判断できます。PKIX では、すべての CRL にこの拡張が必要です。

OID

2.5.29.20

Criticality

この拡張は重要ではありません。

パラメーター

表B.41 CRLNumber 設定パラメーター

パラメーター	説明
enable	ルールが有効であるかどうかを指定します (デフォルト)。
critical	拡張がクリティカルとマークされるかどうかを設定します。デフォルトは非クリティカルです。

B.4.2.1.4. deltaCRLIndicator

deltaCRLIndicator 拡張機能は、デルタ CRL を生成します。これは、最後の CRL 以降に取り消された証明書のみのリストです。また、ベース CRL への参照も含まれています。これにより、ローカルデータベースに既に存在する未変更の情報を無視しながら、ローカルデータベースが更新されます。これにより、失効情報を CRL 構造以外の形式で格納するアプリケーションの処理時間を大幅に改善できます。

OID

2.5.29.27

Criticality

PKIX では、その拡張が存在する場合はこの拡張が必須でなければなりません。

パラメーター

表B.42 DeltaCRL 設定パラメーター

パラメーター	説明
enable	ルールが有効かどうかを指定します。デフォルトでは無効になっています。

パラメーター	説明
critical	拡張機能がクリティカル、または非クリティカルを設定します。デフォルトでは、これはクリティカルになっています。

B.4.2.1.5. FreshestCRL

freshestCRL 拡張機能は、デルタ CRL 情報の取得方法を識別します。freshestCRL 拡張機能は完全な CRL に配置され、最新のデルタ CRL の場所を示します。

OID

2.5.29.46

Criticality

PKIX では、この拡張機能は非クリティカルである必要があります。

パラメーター

表B.43 FreshestCRL 設定パラメーター

パラメーター	説明
enable	拡張機能ルールが有効かどうかを指定します。デフォルトでは、これは無効になっています。
critical	拡張にクリティカルまたは非クリティカルのマークを付けます。デフォルトはクリティカルではありません。
numPoints	デルタ CRL の発行ポイントの数を、**0** から任意の正の整数に指定します。デフォルトは **0** です。これを 0 以外の整数に設定する場合は、数値を設定してから **OK** をクリックしてウィンドウを閉じます。ルールの編集ウィンドウを再度開き、これらのポイントを設定するフィールドが存在するようになります。
pointType**n**	**n** 発行ポイントの発行ポイントのタイプを指定します。**numPoints** で指定する数字ごとに、**pointType** パラメーターの等しい数があります。オプションは **DirectoryName** または **URIName** です。

パラメーター	説明
pointNamen	**pointType** が **directoryName** に設定されている場合、値は証明書のサブジェクト名と同様に、X.500 名の形式の文字列である必要があります。たとえば、**CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 **pointType** が **URIName** に設定されている場合、名前は URI である必要があります。URI は絶対パス名で、ホストを指定する必要があります。例: **http://testCA.example.com/get/crls/here/**。

B.4.2.1.6. issuerAltName

Issuer Alternative Name 拡張機能を使用すると、メールアドレス、DNS 名、IP アドレス (IPv4 と IPv6 の両方)、URI (Uniform Resource Indicator) などのバインディング属性など、CRL の発行者を使用して、追加の ID を CRL の発行者に関連付けることができます。詳細は、「issuerAltName 拡張」 の証明書拡張を参照してください。

OID

2.5.29.18

パラメーター

表B.44 IssuerAlternativeName 設定パラメーター

パラメーター	説明
enable	拡張ルールが有効であるかどうかを設定します。デフォルトでは無効になっています。
critical	拡張がクリティカルかどうかを設定します。デフォルトでは、これは非クリティカルになります。
numNames	拡張で許可される代替名または ID の合計数を設定します。それぞれの名前には、設定パラメーター (**nameType** および **name**) のセットがあります。これには適切な値が必要です。そうでない場合、ルールはエラーを返します。このフィールドで指定された値を変更して、ID の総数を変更します。拡張機能に含めることができる ID の総数に制限はありません。設定パラメーターの各セットは、このフィールドの値から派生した整数によって区別されます。たとえば、**numNames** パラメーターが **2** に設定されている場合、派生整数は **0** と **1** です。

パラメーター	説明
nameType**n**	general-name タイプを指定します。次のいずれかになります。 ● 名前がインターネットメールアドレスの場合は **rfc822Name**。 ● 名前が X.500 ディレクトリー名である場合に **directoryName**。 ● **dNSName** 名前が DNS 名である場合。 ● **ediPartyName** 名前が EDI 当 事名である場合。 ● **URL** 名前が URI (デフォルト) の場合。 ● **iPAddress** 名前が IP アドレスの場合。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 ● 名前がオブジェクト識別子である場合は **OID**。 ● 名前が他の名前形式である場合は **otherName** です。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。
name**n**	一般名の値を指定します。許可される値は、**nameType** フィールドで指定された名前の種類によって異なります。 ● **rfc822Name** の場合は、値が **local-part@domain** 形式の有効なインターネットメールアドレスである必要があります。 ● **directoryName** の場合、この値は証明書のサブジェクト名と同様に X.500 名である必要があります。たとえば、**CN=CACentral,OU=Research Dept,O=Example Corporation,C=US** となります。 ● **dNSName** の場合、値は DNS 形式の有効なドメイン名である必要があります。例: **testCA.example.com**

パラメーター	説明
	• **eDIPartyName** の場合、名前は IA5String である必要があります。例: **Example Corporation**。 • **URL** の場合、値は相対的でない URI である必要があります。例: **http://testCA.example.com**。 • **iPAddress** の場合、値は、ドットで区切られたコンポーネント表記で指定された有効な IP アドレスである必要があります。IP アドレス、またはネットマスクを含む IP アドレスになります。IPv4 アドレスは、**n.n.n.n** または **n.n.n.n,m.m.m.m** の形式にする必要があります。たとえば、**128.21.39.40** または **128.21.39.40,255.255.255.00** です。IPv6 アドレスは 128 ビット名前空間を使用します。IPv6 アドレスはコロンで区切られ、ネットマスクはピリオドで区切られます。たとえば、**0:0:0:0:0:0:13.1.68.3**、**FF01::43**、**0:0:0:0:0:0:13.1.68.3,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.0**、および **FF01::43,FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FF00:0000** になります。 • **OID** の場合、この値は、ドットで区切られた数値コンポーネント表記で指定された一意の有効な OID である必要があります。たとえば、**1.2.3.4.55.6.5.99** です。カスタム OID を使用してサーバーを評価およびテストできますが、実稼働環境では、OID の定義および ID のサブツリーの登録に関する ISO 規則に準拠します。 • **otherName** の場合、名前には他の形式が使用されます。これは、**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any**、および **KerberosName** をサポートします。**PrintableString**、**IA5String**、**UTF8String**、**BMPString**、**Any** は、**/var/lib/pki/pki-tomcat/ca/othername.txt** などのサブツリーをしている base-64 エンコードファイルに文字列を設定します。**KerberosName** には、**realm1|0|userID1,userID2** などの **Realm|NameType|NameStrings** 形式になります。名前は、base-64 でエンコードされた形式の一般名を含むファイルへの絶対パスである必要があります。たとえば、**/var/lib/pki/pki-tomcat/ca/extn/ian/othername.txt** になります。

B.4.2.1.7. issuingDistributionPoint

Issuing Distribution Point CRL 拡張機能は、特定の CRL の CRL ディストリビューションポイントを識別し、エンドエンティティー証明書のみの失効、CA 証明書のみ、または理由コードのセットが制限されている失効証明書など、対象となる失効の種類を示します。

PKIX Part I ではこの拡張は必要ありません。

OID

2.5.29.28

Criticality

PKIX では、その拡張が存在する場合はこの拡張が必須でなければなりません。

パラメーター

表B.45 IssuingDistributionPoint 設定パラメーター

パラメーター	説明
enable	拡張機能を有効にするかどうかを設定します。デフォルトは無効です。
critical	拡張機能をクリティカル、デフォルト、または非クリティカルとしてマークします。
pointType	以下から発行配布ポイントのタイプを指定します。 • **directoryName** は、タイプが X.500 ディレクトリー名であることを指定します。 • **URI** は、タイプが統一されたリソースインジケーターであることを指定します。 • **RelativeToIssuer** は、タイプが **ou=Engineering** などの DN の単一ノードを表す相対識別名 (RDN) であることを指定します。

パラメーター	説明
pointName	発行されたディストリビューションポイントの名前を指定します。分散ポイントの名前は、**pointType** パラメーターに指定された値によって異なります。 ● **directoryName** の場合、名前は X.500 名である必要があります。たとえば、**cn=CRLCentral,ou=Research Dept,o=Example Corporation,c=US** となります。 ● **URIName** では、名前は絶対パス名であり、ホストを指定する URI である必要があります。例: **http://testCA.example.com/get/crls/here/**。  **注記** CRL は、CRL 発行ポイントに対応するディレクトリーエントリーに格納される場合があります。これは、CA のディレクトリーエントリーとは異なる場合があります。
onlySomeReasons	ディストリビューションポイントに関連付けられた理由コードを指定します。 許容値は、理由コード (**unspecified**、**keyCompromise**、**cACompromise**、**affiliationChanged**、**asuperseded**、**cessationOfOperation**、**certificateHold**、および **removeFromCRL**) の組み合わせです。ディストリビューションポイントにすべての理由コード (デフォルト) を含む失効した証明書が含まれている場合は、フィールドを空白のままにします。
onlyContainsCACerts	設定されている場合に限り、ディストリビューションポイントにユーザー証明書が含まれることを指定します。デフォルトでは、これは設定されていません。つまり、ディストリビューションポイントにはすべての種類の証明書が含まれています。
indirectCRL	ディストリビューションポイントに間接 CRL が含まれていることを指定します。デフォルトでは選択されていません。

B.4.2.2. CRL エントリー拡張

次のセクションでは、インターネット X.509v3 公開鍵インフラストラクチャーで提案されている標準の一部として定義されている CRL エントリー拡張タイプを示します。これらの拡張機能はすべてクリティカルではありません。

B.4.2.2.1. certificateIssuer

Certificate Issuer 拡張機能は、間接 CRL のエントリーに関連付けられている証明書発行者を識別します。

この拡張機能は、Certificate System でサポートされていない間接 CRL でのみ使用されます。

OID

2.5.29.29

B.4.2.2.2. invalidityDate

Invalidity Date 拡張機能は、秘密鍵が侵害された日付、または証明書が無効になった日付を提供します。

OID

2.5.29.24

パラメーター

表B.46 InvalidityDate 設定パラメーター

パラメーター	説明
enable	拡張機能ルールを有効にするか無効にするかを設定します。デフォルトでは、これは有効になります。
critical	拡張機能をクリティカルまたは非クリティカルとしてマークします。デフォルトでは、これは非クリティカルではありません。

B.4.2.2.3. CRLReason

Reason Code 拡張は、証明書失効リストの理由を識別します。

OID

2.5.29.21

パラメーター

表B.47 CRLReason 設定パラメーター

パラメーター	説明
enable	拡張機能ルールを有効にするか無効にするかを設定します。デフォルトでは、これは有効になります。
critical	拡張にクリティカルまたは非クリティカルのマークを付けます。デフォルトでは、これはクリティカルではありません。

B.4.3. Netscape で定義された証明書拡張のリファレンス

Netscape は、その製品に対して特定の証明書拡張を定義しました。一部の拡張機能は現在廃止されており、その他の拡張機能は X.509 提案標準で定義されている拡張機能に取って代わられています。すべての Netscape 拡張機能は、証明書に存在することでその証明書が他のクライアントと互換性がなくなることがないように、非クリティカルなものとしてタグ付けする必要があります。

B.4.3.1. netscape-cert-type

Netscape Certificate Type 拡張機能を使用して、証明書を使用できる目的を制限できます。これは、X.509 v3 拡張 「extKeyUsage」 および 「basicConstraints」 に置き換えられています。

拡張機能が証明書に存在する場合は、指定した使用に対して証明書を制限します。拡張機能が存在しない場合、証明書はオブジェクト署名を除くすべてのアプリケーションで使用できます。

値はビット文字列であり、個々のビット位置が設定されると、次のように特定の用途のために証明書を認証します。

- ビット 0 - SSL クライアント証明書
- ビット 1 - SSL サーバー証明書
- ビット 2 - S/MIME 証明書
- ビット 3 - オブジェクト署名証明書
- ビット 4 - 予約
- ビット 5 - SSL CA 証明書
- ビット 6 - S/MIME CA 証明書
- ビット 7 - オブジェクト署名 CA 証明書

OID

2.16.840.1.113730.1.1

B.4.3.2. netscape-comment

この拡張機能の値は IA5String です。これは、証明書を表示する際にユーザーに表示されるコメントです。

OID

2.16.840.1.113730.13

付録C 公開モジュールのリファレンス

いくつかのパブリッシャー、マッパー、およびルールモジュールは、デフォルトで Certificate Manager を使用して設定されます。

- 「パブリッシャープラグインモジュール」
- 「マッパープラグインモジュール 」
- 「ルールインスタンス」

C.1. パブリッシャープラグインモジュール

このセクションでは、Certificate Manager に提供されるパブリッシャーモジュールを説明します。これらのモジュールは Certificate Manager で使用され、特定のパブリッシャーインスタンスを有効および設定します。

- 「FileBasedPublisher」
- 「LdapCaCertPublisher」
- 「LdapUserCertPublisher」
- 「LdapCrlPublisher」
- 「LdapDeltaCrlPublisher」
- 「LdapCertificatePairPublisher」
- 「OCSPPublisher」

C.1.1. FileBasedPublisher

FileBasedPublisher プラグインモジュールは、証明書および CRL をファイルに公開するように Certificate Manager を設定します。このプラグインは、発行元の設定時に選択したチェックボックスに応じて、base-64 でエンコードされたファイル、DER でエンコードされたファイル、またはその両方を発行できます。証明書および CRL の内容は、**PrettyPrintCert** および **PrettyPrintCRL** ツールを使用してファイルを変換して表示できます。base-64 および DER でエンコードされた証明書および CRL でコンテンツを表示する方法は、「ファイルに公開される証明書および CRL の表示」 を参照してください。

デフォルトでは、Certificate Manager は **FileBasedPublisher** モジュールのインスタンスを作成しません。

表C.1 FileBasedPublisher 設定パラメーター

パラメーター	説明
Publisher ID	パブリッシャーの名前、スペースを含まない英数字の文字列を指定します。例: **PublishCertsToFile**

パラメーター	説明
directory	Certificate Manager がファイルを作成するディレクトリーへの完全なパスを指定します。パスは絶対パスにすることも、Certificate System インスタンスディレクトリーからの相対パスにすることもできます。たとえば、**/export/CS/certificates** です。

C.1.2. LdapCaCertPublisher

LdapCaCertPublisher プラグインモジュールは、CA 証明書の CA 証明書を CA のディレクトリーエントリーの **caCertificate;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

モジュールは、CA エントリーのオブジェクトクラスを pkiCA または certificationAuthority に変換していない場合はこれを **pkiCA** または **certificationAuthority** に変換します。同様に、CA に他の証明書がない場合に、**pkiCA** または **certificationAuthority** オブジェクトクラスも削除します。

インストール中に、Certificate Manager は、CA 証明書をディレクトリーに発行するための **LdapCaCertPublisher** モジュールのインスタンスを自動的に作成します。

表C.2 LdapCaCertPublisher 設定パラメーター

パラメーター	説明
caCertAttr	CA 証明書を公開する LDAP ディレクトリー属性を指定します。これは、**caCertificate;binary** でなければなりません。
caObjectClass	ディレクトリー内の CA のエントリーのオブジェクトクラスを指定します。これは **pkiCA** または **certificationAuthority** でなければなりません。

C.1.3. LdapUserCertPublisher

LdapUserCertPublisher プラグインモジュールは、User 証明書の User 証明書を ユーザーのディレクトリーエントリーの **userCertificate;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

このモジュールは、エンドエンティティー証明書を LDAP ディレクトリーに公開するために使用されます。エンドエンティティーの証明書のタイプには、SSL クライアント、S/MIME、SSL サーバー、および OCSP レスポンダーが含まれます。

インストール中、Certificate Manager は、ディレクトリーにエンドエンティティー証明書を発行するための **LdapUserCertPublisher** モジュールのインスタンスを自動的に作成します。

表C.3 LdapUserCertPublisher 設定パラメーター

パラメーター	説明

パラメーター	説明
certAttr	Certificate Manager が証明書を公開するマップされたエントリーのディレクトリー属性を指定します。これは **userCertificate;binary** である必要があります。

C.1.4. LdapCrlPublisher

LdapCrlPublisher プラグインモジュールは、CRL をディレクトリーエントリーの **certificateRevocationList;binary** 属性に公開または公開しないように、証明書マネージャーを設定します。

インストール中に、Certificate Manage は、自動的に CA 証明書をディレクトリーに公開するための **LdapCrlPublisher** モジュールのインスタンスを作成します。

表C.4 LdapCrlPublisher 設定パラメーター

パラメーター	説明
crlAttr	証明書マネージャーが CRL を公開するマップされたエントリーのディレクトリー属性を指定します。これは **certificateRevocationList;binary** でなければなりません。

C.1.5. LdapDeltaCrlPublisher

LdapDeltaCrlPublisher プラグインモジュールは、Delta CRL をディレクトリーエントリーの **deltaRevocationList** 属性に公開または公開しないように、証明書マネージャーを設定します。

インストール中に、Certificate Manage は、自動的に CA 証明書をディレクトリーに公開するための **LdapDeltaCrlPublisher** モジュールのインスタンスを作成します。

表C.5 LdapDeltaCrlPublisher 設定パラメーター

パラメーター	説明
crlAttr	Certificate Manager がデルタ CRL を公開するマップされたエントリーのディレクトリー属性を指定します。これは **deltaRevocationList;binary** である必要があります。

C.1.6. LdapCertificatePairPublisher

LdapCertificatePairPublisher プラグインモジュールは、CA のディレクトリーエントリーの **crossCertPair;binary** 属性に、クロス署名証明書を公開または公開しないように証明書マネージャーを設定します。

モジュールは、CA エントリーのオブジェクトクラスを pkiCA または certificationAuthority に変換していない場合はこれを **pkiCA** または **certificationAuthority** に変換します。同様に、CA に他の証明書がない場合に、**pkiCA** または **certificationAuthority** オブジェクトクラスも削除します。

インストール時に、証明書マネージャーは **LdapCrossCertPairPublisher** という名前の **LdapCertificatePairPublisher** モジュールのインスタンスを作成し、複数の署名された証明書をディレクトリーに公開します。

表C.6 LdapCertificatePairPublisher パラメーター

パラメーター	説明
crossCertPairAttr	CA 証明書を公開する LDAP ディレクトリー属性を指定します。これは **crossCertificatePair;binary** でなければなりません。
caObjectClass	ディレクトリー内の CA のエントリーのオブジェクトクラスを指定します。これは **pkiCA** または **certificationAuthority** でなければなりません。

C.1.7. OCSPPublisher

OCSPPublisher プラグインモジュールは、CRL を Online Certificate Status Manager に公開するように Certificate Manager を設定します。

Certificate Manager は、インストール時に **OCSPPublisher** モジュールのインスタンスを作成しません。

表C.7 OCSPPublisher パラメーター

パラメーター	説明
host	Online Certificate Status Manager の完全修飾ホスト名を指定します。
ポート	Online Certificate Status Manager が Certificate Manager をリッスンしているポート番号を指定します。これは Online Certificate Status Manager の SSL ポート番号です。
path	CRL を公開するためのパスを指定します。これはデフォルトのパス **/ocsp/agent/ocsp/addCRL** でなければなりません。
enableClientAuth	クライアント (証明書ベースの) 認証を使用して OCSP サービスにアクセスするかどうかを設定します。
ニックネーム	クライアント認証に使用する OCSP サービスのデータベース内の証明書のニックネームを指定します。これは、**enableClientAuth** オプションが true に設定されている場合にのみ使用されます。

C.2. マッパープラグインモジュール

このセクションでは、Certificate Manager に提供されるマッパープラグインモジュールを説明します。これらのモジュールは、特定のマッパーインスタンスを有効化および設定するように Certificate Manager を設定します。

利用可能なマッパープラグインモジュールには以下が含まれます。

- 「LdapCaSimpleMap」
- 「LdapDNExactMap」
- 「LdapSimpleMap」
- 「LdapSubjAttrMap」
- 「LdapDNCompsMap」

C.2.1. LdapCaSimpleMap

LdapCaSimpleMap プラグインモジュールは、LDAP ディレクトリーに CA のエントリーを自動的に作成し、証明書要求、証明書サブジェクト名、証明書拡張子で指定されたコンポーネント、および属性変数アサーション (AVA) 定数からエントリーの DN を作成することにより、CA の証明書をディレクトリーエントリーにマップするように Certificate Manager を設定します。AVA の詳細は、ディレクトリーのドキュメントを参照してください。

CA 証明書マッパーは、CA のエントリーを作成するか、証明書を既存のエントリーにマップするか、またはその両方を行うかを指定します。

公開ディレクトリーに CA エントリーがすでに存在し、このマッパーの **dnPattern** パラメーターに割り当てられた値が変更しても、**uid** 属性および **o** 属性が同じである場合、マッパーは 2 つ目の CA エントリーの作成に失敗します。たとえば、ディレクトリーに **uid=CA,ou=Marketing,o=example.com** の CA エントリーがすでにあり、**uid=CA,ou=Engineering,o=example.com** で別の CA エントリーを作成するようマッパーが設定されている場合、操作は失敗します。

ディレクトリーの *UID Uniqueness* プラグインが特定のベース DN に設定されているため、操作が失敗する場合があります。この設定により、ディレクトリーがそのベース DN の下に同じ UID を持つ 2 つのエントリーを持つことを防ぎます。この例では、ディレクトリーが **o=example.com** の下に同じ UID **CA** を持つ 2 つのエントリーを持つことを防ぎます。

マッパーが 2 番目の CA エントリーの作成に失敗した場合は、UID 一意性プラグインが設定されているベース DN を確認し、同じ UID のエントリーがディレクトリーにすでに存在するかどうかを確認します。必要に応じて、マッパー設定を調整するか、古い CA エントリーを削除するか、プラグインをコメントアウトするか、エントリーを手動で作成します。

インストール時に、Certificate Manager は CA 証明書マッパーモジュールの 2 つのインスタンスを自動的に作成します。マッパーの名前は以下のように命名されます。

- CRL の **LdapCrlMap** (「LdapCrlMap」 を参照)
- CA の **LdapCaCertMap** (「LdapCaCertMap」 を参照)

表C.8 LdapCaSimpleMap 設定パラメーター

パラメーター	説明

パラメーター	説明
createCAEntry	選択した場合は CA のエントリーを作成します (デフォルト)。 選択した場合、Certificate Manager は最初にディレクトリーに CA のエントリーを作成しようとします。Certificate Manager がエントリーの作成に成功すると、CA の証明書をエントリーに公開しようとします。このチェックボックスを選択しないと、公開するためにエントリーがすでに存在している必要があります。
dnPattern	Certificate Manager が公開ディレクトリーで CA のエントリーを検索するために構築するために使用する DN パターンを指定します。**dnPattern** の値は、コンマで区切られた AVAs の一覧です。AVA は、Certificate Manager が、(**cn=$subj.cn** など) の証明書のサブジェクト名または制約から派生する変数 (**o=Example Corporation** など) です。 CA 証明書に **cn** コンポーネントがない場合に、サブジェクト名のコンポーネントで、CA 証明書マッピングの DN パターンを調整して、CA 証明書が公開されるディレクトリー内のエントリーの DN を反映します。たとえば、CA 証明書のサブジェクト DN が **o=Example Corporation** で、ディレクトリーの CA エントリーが **cn=Certificate Authority, o=Example Corporation** の場合、パターンは **cn=Certificate Authority, o=$subj.o** になります。 ● 例 1: **uid=CertMgr, o=Example Corporation** ● 例 2: **cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US** ● 例 3: **uid=$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou** 上記の例では、**$req** は証明書要求の属性、**$subj** は証明書のサブジェクト名の属性、**$ext** は証明書の拡張子の属性を取ります。

C.2.1.1. LdapCaCertMap

LdapCaCertMap マッパーは、**LdapCaSimpleMap** モジュールのインスタンスです。Certificate Manager は、インストール時にこのマッパーを自動的に作成します。

このマッパーは、ディレクトリーに CA のエントリーを作成し、CA 証明書をディレクトリー内の CA のエントリーを作成します。

デフォルトでは、マッパーはディレクトリーに CA のエントリーを作成するように設定されています。CA のエントリーを見つけるためのデフォルトの DN パターンは次のとおりです。

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.1.2. LdapCrlMap

LdapCrlMap マッパーは **LdapCaSimpleMap** モジュールのインスタンスです。Certificate Manager は、インストール時にこのマッパーを自動的に作成します。

このマッパーは、ディレクトリー内に CA のエントリーを作成し、CRL をディレクトリー内の CA のエントリーにマップします。

デフォルトでは、マッパーは、ディレクトリーに CA のエントリーを作成するように設定されます。CA のエントリーを見つけるデフォルトの DN パターンは以下のとおりです。

```
uid=$subj.cn,ou=people,o=$subj.o
```

C.2.2. LdapDNExactMap

LdapDNExactMap プラグインモジュールは、証明書のサブジェクト名と一致する LDAP エントリー DN を検索することにより、証明書を LDAP ディレクトリーエントリーにマップするように Certificate Manager を設定します。このマッパーを使用するには、各証明書のサブジェクト名がディレクトリーエントリーの DN と完全に一致する必要があります。たとえば、証明書サブジェクト名が **uid=jdoe, o=Example Corporation, c=US** の場合には、証明書マネージャーは **DN uid=jdoe, o=Example Corporation, c=US** を持つエントリーのみを検索します。

一致するエントリーが見つからない場合、サーバーはエラーを返し、証明書を公開しません。

このマッパーは、証明書からすべての値を取得するため、パラメーターに値を必要としません。

C.2.3. LdapSimpleMap

LdapSimpleMap プラグインモジュールは、証明書要求、証明書のサブジェクト名、証明書の拡張子、および属性変数アサーション (AVA) 定数で指定されたコンポーネントからエントリーの DN を引き出すことで、証明書を LDAP ディレクトリーエントリーにマッピングするように証明書マネージャーを設定します。AVA の詳細は、ディレクトリーのドキュメントを参照してください。

デフォルトでは、Certificate Manager は単純なマッパーに基づくマッパールールを使用します。インストール中に、Certificate Manager は、**LdapUserCertMap** という名前が付けられた単純なマッパーモジュールのインスタンスを自動的に作成します。デフォルトのマッパーは、さまざまなタイプのエンドエンティティー証明書を対応するディレクトリーエントリーにマップします。

シンプルなマッパーには 1 つのパラメーター **dnPattern** が必要です。**dnPattern** の値は、コンマで区切られた AVAs の一覧です。AVA は、**uid=$subj.UID** などの変数や、**o=Example Corporation** などの定数になります。

- 例 1: **uid=CertMgr, o=Example Corporation**
- 例 2: **cn=$subj.cn,ou=$subj.ou,o=$subj.o,c=US**
- 例 3: uid=**$req.HTTP_PARAMS.uid, e=$ext.SubjectAlternativeName.RFC822Name,ou=$subj.ou**

例では、**$req** は、証明書要求から属性を取得し、**$subj** は、証明書のサブジェクト名から属性を取得し、**$ext** は、証明書拡張から属性を取得します。

C.2.4. LdapSubjAttrMap

LdapSubjAttrMap プラグインモジュールは、設定可能な LDAP 属性を使用して証明書を LDAP ディレクトリーエントリーにマップするように Certificate Manager を設定します。このマッパーを使用するには、ディレクトリーエントリーに指定された LDAP 属性を含める必要があります。

Certificate Manager は、サブジェクト DN 全体と完全に一致する値を持つ属性をディレクトリーで検索するため、このマッパーにはサブジェクト DN の正確なパターンが必要です。たとえば、指定された LDAP 属性が **certSubjectDN** で、証明書サブジェクト名が **uid=jdoe, o=Example Corporation, c=US** の場合、証明書マネージャーは、**certSubjectDN=uid=jdoe** 属性のあるエントリーに対してディレクトリーを検索します。

一致するエントリーが見つからない場合、サーバーはエラーを返し、ログに書き込みます。

表C.9「LdapSubjAttrMap Parameters」 これらのパラメーターを説明します。

表C.9 LdapSubjAttrMap Parameters

パラメーター	説明
certSubjNameAttr	証明書のサブジェクト名を値として含む LDAP 属性の名前を指定します。デフォルトは **certSubjectName** ですが、任意の LDAP 属性に設定できます。
searchBase	属性検索を開始するベース DN を指定します。許容値は、**o=example.com, c=US** などの LDAP エントリーの有効な DN です。

C.2.5. LdapDNCompsMap

LdapDNCompsMap プラグインモジュールは、DN コンポーネントマッパーを実装します。このマッパーは、証明書のサブジェクト名で指定された **cn** コンポーネント、**ou** コンポーネント、**o** コンポーネント、および **c** コンポーネントからエントリーの DN を構築し、それを検索 DN として使用してディレクトリー内のエントリーを検索することで、証明書を LDAP ディレクトリーエントリーにマッピングします。マッパーは以下のエントリーを見つけます。

- CA 証明書と CRL を公開するためのディレクトリー内の CA のエントリー。
- エンドエンティティー証明書を公開するためのディレクトリーのエンドエンティティー。

マッパーは DN コンポーネントを取得して検索 DN を構築します。マッパーは任意の root 検索 DN も取得します。サーバーは DN コンポーネントを使用して LDAP エントリーを形成し、サブツリー検索を開始し、フィルターコンポーネントを使用してサブツリーの検索フィルターを形成します。DN コンポーネントが設定されていないと、サーバーはサブツリーにベース DN を使用します。ベース DN が null で、DN コンポーネントが一致しない場合には、エラーが返されます。DN コンポーネントおよびフィルターコンポーネントがいずれも一致しない場合は、エラーが返されます。フィルターコンポーネントが null の場合は、ベース検索が実行されます。

DNComps パラメーターおよび **filterComps** パラメーターは、有効な DN コンポーネントまたは属性をコンマで区切って指定します。パラメーターは、属性の複数のエントリーを受け入れません。たとえば、**filterComps** を **cn,ou** に設定することができますが、**cn,ou2ou1** には設定できません。ディレクトリーエントリーに複数の **ou** が含まれている場合など、同じ属性の複数のインスタンスを持つフィルターを作成するには、**LdapDNCompsMap** モジュールのソースコードを修正します。

以下のコンポーネントは、DN で一般的に使用されます。

- **uid** は、ディレクトリー内のユーザーのユーザー ID を表します。
- **cn** は、ディレクトリー内のユーザーの共通名を表します。
- **ou** は、ディレクトリー内の組織単位を表します。
- **o** は、ディレクトリー内の組織を表します。
- **l** は地域 (都市) を表します。
- **st** は状態を表します。
- **c** は国を表します。

たとえば、次の DN は、米国カリフォルニア州マウンテンビューにある Example Corporation の営業部門で働く Jane Doe というユーザーを表しています。

```
cn=Jane Doe, ou=Sales, o=Example Corporation, l=Mountain View, st=California, c=US
```

証明書マネージャーは、コンポーネント (**cn**、**ou**、**o**、**l**、**st**、および **c**) の一部またはすべてを使用して、ディレクトリーを検索するための DN を構築します。マッパールールを作成するときに、これらのコンポーネントをサーバーが DN の構築に使用するように指定できます。つまり、ディレクトリー内の属性に一致するコンポーネントです。これは **dnComps** パラメーターで設定されます。

たとえば、コンポーネント **cn**、**ou**、**o**、および **c** は、**dnComps** パラメーターの値として 設定されます。ディレクトリー内の Jane Doe のエントリーを見つけるために、Certificate Manager は、証明書から DN 属性値を読み取ることによって次の DN を構築し、ディレクトリーを検索するためのベースとして DN を使用します。

```
cn=Jane Doe, ou=Sales, o=Example Corporation, c=US
```

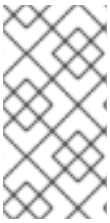
- サブジェクト名には、**dnComps** パラメーターで指定されたすべてのコンポーネントを含める必要はありません。この例の **l** および **st** など、サーバーは、サブジェクト名の一部ではないコンポーネントを無視します。
- 未指定のコンポーネントは、DN の構築には使用されません。例では、**ou** コンポーネントが含まれていない場合、サーバーはこの DN をディレクトリー検索のベースとして使用します。

  ```
  cn=Jane Doe, o=Example Corporation, c=US
  ```

dnComps パラメーターの場合、Certificate Manager が LDAP DN を正確に形成するために使用できる DN コンポーネントを入力します。ただし、特定の状況では、証明書のサブジェクト名がディレクトリー内の複数のエントリーと一致する場合があります。次に、Certificate Manager は、DN から一致するエントリーを 1 つ取得できない可能性があります。たとえば、サブジェクト名 **cn=Jane Doe, ou=Sales, o=Example Corporation, c=US** は、ディレクトリー内の Jane Doe という名前の 2 つのユーザーと一致する可能性があります。その場合、Certificate Manager には、証明書のサブジェクトに対応するエントリーを決定するための追加の基準が必要です。

Certificate Manager がディレクトリー内の異なるエントリーを区別するために使用する必要のあるコンポーネントを指定するには、**filterComps** パラメーターを使用します。詳細は、表 C.10「LdapDNCompsMap 設定パラメーター」 を参照してください。たとえば、**cn**、**ou**、**o**、および **c** が**dnComps** パラメーターの値である場合、**l** 属性を使用して同じ **cn**、**ou**、**o**、および **c** 値を持つエントリーを区別できる場合にのみ、**filterComps** パラメーター に **l** を入力します。

2 つの Jane Doe エントリーが **uid** 属性の値によって区別される場合、(1 つのエントリー**uid** は **janedoe1** で、(もう 1 つのエントリーの **uid** は **janedoe2**)、証明書のサブジェクト名を設定して **uid** コンポーネントを含めるようにできます。



注記

e、**l**、および **st** コンポーネントは、終了エンティティー向けに提供される証明書要求の標準フォームに含まれません。これらのコンポーネントをフォームに追加することも、証明書発行フォームのサブジェクト名を編集するときに発行エージェントにこれらのコンポーネントの挿入を要求することもできます。

C.2.5.1. LdapDNCompsMap の設定パラメーター

この設定では、Certificate Manager は、DN を形成するための **dnComps** 値と、サブツリーの検索フィルターを形成するための **filterComps** 値を使用して、L その証明書と、DAP ディレクトリー内の証明書をマップします。

- フォーム化された DN が null の場合、サーバーはサブツリーの **baseDN** の値を使用します。正式な DN とベース DN の両方が null の場合、サーバーはエラーをログに記録します。
- フィルターが null の場合、サーバーは検索に **baseDN** の値を使用します。フィルターとベース DN の両方が null の場合、サーバーはエラーをログに記録します。

表C.10「LdapDNCompsMap 設定パラメーター」 これらのパラメーターを説明します。

表C.10 LdapDNCompsMap 設定パラメーター

パラメーター	説明
baseDN	公開ディレクトリー内のエントリーの検索を開始する DN を指定します。**dnComps** フィールドが空の場合、サーバーはベース DN 値を使用してディレクトリーの検索を開始します。
dnComps	公開ディレクトリーのどこで、Certificate Manager が CA またはエンドエンティティーの情報と一致する LDAP エントリーの検索を開始するかを指定します。 たとえば、**dnComps** が、DN の **o** 属性および **c** 属性を使用する場合、サーバーは、ディレクトリーで **o=org**、**c=country** エントリーから検索を開始し、**org** と **country** を証明書の DN の値に置き換えます。 **dnComps** フィールドが空の場合、サーバーは、**baseDN** フィールドを確認し、**filterComps** パラメーター値により指定されるフィルターに一致するエントリーの DN によって指定されるディレクトリーツリーを検索します。 許容値は、有効な DN コンポーネントまたは属性をコンマで区切って指定します。

パラメーター	説明
filterComps	Certificate Manager が検索結果からエントリーをフィルタリングするために使用するコンポーネントを指定します。サーバーは **filterComps** 値を使用して、サブツリーの LDAP 検索フィルターを形成します。サーバーは、証明書のサブジェクト名からこれらの属性の値を収集することにより、フィルターを構築します。フィルターを使用して、LDAP ディレクトリー内のエントリーを検索して照合します。 サーバーが証明書から収集した情報と一致するエントリーをディレクトリー内で複数検出した場合、検索は成功し、サーバーはオプションで検証を実行します。たとえば、**filterComps** が電子メールおよびユーザー ID 属性を使用するように設定されていると (**filterComps=e,uid**)、サーバーはディレクトリーを検索して、電子メールとユーザー ID の値が証明書から収集された情報と一致するエントリーを検索します。 許容値は、コンマで区切られた証明書 DN 内の有効なディレクトリー属性です。フィルターの属性名は、LDAP ディレクトリー内の属性名ではなく、証明書の属性名である必要があります。たとえば、ほとんどの証明書には、ユーザーの電子メールアドレスの **e** 属性があります。LDAP は、その属性の **mail** を呼び出します。

C.3. ルールインスタンス

このセクションでは、設定したルールインスタンスを説明します。

C.3.1. LdapCaCertRule

LdapCaCertRule は、CA 証明書を LDAP ディレクトリーに公開するために使用することができます。

表C.11 LdapCaCert Rule 設定パラメーター

パラメーター	値	説明
type	**cacert**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCaCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCaCertMap」を参照してください。

パラメーター	値	説明
publisher	**LdapCaCertPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapCaCertPublisher」 を参照してください。

C.3.2. LdapXCertRule

LdapXCertRule は、LDAP ディレクトリーにコピー間の証明書を公開するのに使用されます。

表C.12 LdapXCert Rule 設定パラメーター

パラメーター	値	説明
type	**xcert**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCaCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCaCertMap」を参照してください。
publisher	**LdapCrossCertPairPublisher**	ルールで使用するパブリッシャーを指定します。このパブリッシャーのの詳細は、「LdapCertificatePairPublisher」 を参照してください。

C.3.3. LdapUserCertRule

LdapUserCertRule は、ユーザー証明書を LDAP ディレクトリーに公開するために使用されます。

表C.13 LdapUserCert Rule 設定パラメーター

パラメーター	値	説明
type	**certs**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。

パラメーター	値	説明
enable	**はい**	ルールを有効にします。
mapper	**LdapUserCertMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapSimpleMap」を参照してください。
publisher	**LdapUserCertPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapUserCertPublisher」を参照してください。

C.3.4. LdapCRLRule

LdapCRLRule は CRL を LDAP ディレクトリーに公開するために使用されます。

表C.14 LdapCRL ルール設定パラメーター

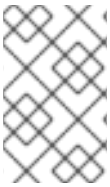
パラメーター	値	説明
type	**crl**	公開される証明書のタイプを指定します。
predicate		パブリッシャーの述語を指定します。
enable	**はい**	ルールを有効にします。
mapper	**LdapCrlMap**	ルールで使用するマッパーを指定します。マッパーの詳細は、「LdapCrlMap」を参照してください。
publisher	**LdapCrlPublisher**	ルールで使用するパブリッシャーを指定します。パブリッシャーの詳細は、「LdapCrlPublisher」を参照してください。

付録D ACL リファレンス

このセクションでは、各リソースが制御するものについて説明し、それらの操作の結果を説明する可能な操作をリストし、定義された各 ACL リソースのデフォルトの ACI を提供します。各サブシステムには、そのサブシステムに関連する ACL のみが含まれます。

D.1. ACL 設定ファイルについて

アクセス制御 は、サーバーの一部にアクセスできるユーザーと、ユーザーが実行できる操作に関するルールを設定する方法です。LDAP ディレクトリーサービスに依存し、Java コンソールを使用する 4 つのサブシステム (CA、KRA、OCSP、および TKS) はすべて、LDAP スタイルのアクセス制御を実装してリソースにアクセスします。これらのアクセス制御リスト (ACL) は、**/var/lib/pki/*instance_name*/conf/*subsystem*/acl.ldif** ファイルにあります。



注記

本セクションでは、アクセス制御の概念の概要を説明します。アクセス制御は、『Red Hat Directory Server 管理ガイド』 の 『アクセス制御の管理』 の章で詳しく説明されています。

Certificate System ACL ファイルは、内部データベースにより読み込まれる LDIF ファイルです。個々の ACL は、***resourceACLS*** 属性として定義されます。これは、保護されているサブシステムの領域を識別する属性と、その後設定されているすべての特定のアクセス制御のリストで識別されます。

```
resourceACLS: class_name:all rights: allow|deny (rights) type=target description
```

リソースへのアクセスを許可または拒否する各ルールは、**アクセス制御命令** (ACI) と呼ばれます。(リソースの ACI の合計はアクセス制御リストです。) 実際の ACI を定義する前に、ACL 属性は、Certificate System サブシステムによって使用される特定のプラグインクラスに最初に適用されます。これにより、各 ACL がサブシステムによって実行される特定の機能に集中し、インスタンスのセキュリティーが強化され、ACL の適用をより適切に制御できるようになります。

例D.1 証明書プロファイルの一覧を表示するデフォルト ACL

```
resourceACLS: certServer.ca.profiles:list:allow (list) group="Certificate Manager
Agents":Certificate Manager agents may list profiles
```

各サブシステム (CA、KRA、OCSP、および TKS) には操作用の異なるリソースがあるため、各サブシステムインスタンスには独自の **acl.ldif** ファイルと独自に定義された ACL があります。

各 ACI は、実行できるアクセスまたは動作 (**右**)、と ACI の適用対象 (**ターゲット**) を定義します。ACI の基本的な形式は次のとおりです。

```
allow|deny (rights) user|group
```

権限 は、ACI がユーザーに実行を許可する操作のタイプです。LDAP ACI の場合は、検索、読み取り、書き込み、削除など、ディレクトリーエントリーに対する権限のリストは比較的限られています。Certificate System は、取り消し、送信、割り当てなどの一般的な PKI タスクを扱う追加の権限を使用します。

操作が ACI で明示的に許可されていない場合、この操作は暗黙的に拒否されます。1 つの ACI で操作が明示的に拒否された場合は、明示的に許可されている ACI よりも優先されます。拒否ルールは、ルールが追加のセキュリティーを提供できるようにするために常に優れています。

各 ACI は特定のユーザーまたはグループに適用する必要があります。エントリーベースのアクセスではなくクライアントベースのアクセスを定義する **ipaddress=** などのオプションがありますが、これは、いくつかの一般的な条件 (通常 **user=** または **group=**) を使用して設定されます。複数の条件がある場合は、論理和 ("or") を表す二重パイプ (||) 演算子と、論理積 ("and") を表す二重アンパサンド (&&) 演算子を使用して条件を設定することができます。たとえば、**group="group1"** || **"group2"** となります。

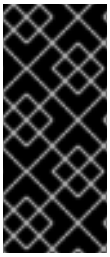
resourceACLS 属性値の各領域は、 表D.1「ACL 属性値のセクション」 で定義されます。

表D.1 ACL 属性値のセクション

値	説明
class_name	ACI が適用されるプラグインクラス。
すべての操作	ACI 定義に含まれるすべての操作の一覧。1 つの ACI には複数の操作があり、1 つの ***resourceACLS*** 属性に複数の ACI があります。
allow|deny	アクションがターゲットユーザーまたはグループに対して許可されているか、ターゲットユーザーまたはグループに対して拒否されているか。
(operations)	許可または拒否されている操作。
type=target	これが適用されるユーザーを特定するターゲット。これは一般的にユーザー (**user=**"name" など) またはグループ (**group=**"group" など) です。条件が複数ある場合は、二重パイプ (||) 演算子 (論理 "or") と二重アンパサンド (&&) 演算子 (論理 "and") を使用して条件を設定することができます。たとえば、**group="group1"** || **"group2"** となります。
説明	ACL が実行していることの説明。

D.2. 共通 ACL

このセクションでは、4 つのサブシステムタイプすべてに共通するデフォルトのアクセス制御設定を説明します。これらのアクセス制御ルールは、ユーザーやグループのログ記録や追加など、基本的で一般的な設定設定へのアクセスを管理します。



重要

これらの ACL は、各サブシステムインスタンスの **acl.ldif** ファイルで同じ ACL が発生するのが一般的です。これらは、設定ファイルまたは設定がすべてのサブシステムインスタンスによって共通に保持されるという意味で、**共有** ACL ではありません。他のすべてのインスタンス設定と同様に、これらの ACL は、インスタンス固有の **acl.ldif** ファイルで、他のサブシステムインスタンスとは独立して維持されます。

D.2.1. certServer.acl.configuration

ACL 設定への操作を制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.2 certServer.acl.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ACL リソースを表示し、ACL リソース、ACL リストエバリュエーター、および ACL エバリュエータータイプを一覧表示します。	許可	管理者 エージェント 監査者
modify	ACL エバリュエーターの追加、削除、および更新。	許可	管理者

D.2.2. certServer.admin.certificate

Certificate Manager から証明書をインポートするユーザーを制御します。デフォルトでは、この操作はすべてのユーザーが許可されます。デフォルト設定は以下のようになります。

```
allow (import) user="anybody"
```

注記

このエントリーは、インスタンスの設定に使用される CA 管理 Web インターフェイスに関連付けられています。この ACL は、インスタンスの設定中にのみ使用可能であり、CA の実行後には使用できません。

表D.3 certServer.admin.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
import	CA 管理者証明書をインポートして、シリアル番号で証明書を取得します。	許可	全ユーザー

D.2.3. certServer.auth.configuration

認証設定の操作を制御します。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators
```

表D.4 certServer.auth.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	認証プラグイン、認証タイプ、設定済みの認証マネージャープラグイン、および認証インスタンスを表示します。認証マネージャープラグインおよび認証マネージャーインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	認証プラグインおよび認証インスタンスを追加または削除します。認証インスタンスを変更します。	許可	管理者

D.2.4. certServer.clone.configuration

クローン作成で使用される設定情報を読み取り、変更できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" || group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators"
```

表D.5 certServer.clone.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	元のインスタンス設定を表示します。	許可	エンタープライズ管理者
modify	元のインスタンス設定を変更します。	許可	エンタープライズ管理者

D.2.5. certServer.general.configuration

CA の設定を表示および編集できるユーザーなど、サブシステムインスタンスの一般的な設定へのアクセスを制御します。

```
allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" ||
group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online
Certificate Status Manager Agents";allow (modify) group="Administrators"
```

表D.6 certServer.general.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	管理用の運用環境、LDAP 設定、SMTP 設定、サーバー統計、暗号化、トークン名、証明書のサブジェクト名、証明書のニックネーム、サーバーによって読み込むすべてのサブシステム、CA 証明書、およびすべての証明書を表示します。	許可	管理者 エージェント 監査者
modify	LDAP データベース、SMTP、および暗号化の設定を変更します。インポート証明書の発行、証明書のインストール、CA 証明書の信頼と信頼解除、クロスペア証明書のインポート、および証明書の削除。サーバーの再起動および停止操作を実行します。すべてのトークンにログインして、トークンのステータスを確認します。オンデマンドでセルフテストを実行します。証明書情報を取得します。証明書サブジェクト名を処理します。証明書サブジェクト名、証明書キーの長さ、および証明書拡張機能を検証します。	許可	管理者

D.2.6. certServer.log.configuration

ログ設定の変更など、Certificate Manager のログ設定へのアクセスを制御します。

```
allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" ||
group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online
Certificate Status Manager Agents";allow (modify) group="Administrators"
```

表D.7 certServer.log.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログプラグインの情報、ログプラグイン設定、およびログインスタンス設定を表示します。ログプラグインおよびログインスタンスを一覧表示します (NTpixel を除く)。	許可	管理者 エージェント 監査者
modify	ログプラグインおよびログインスタンスを追加し、削除します。ログロールオーバーパラメーターやログレベルなど、ログインスタンスを変更します。	許可	管理者

D.2.7. certServer.log.configuration.fileName

インスタンスのログのファイル名を変更するアクセスを制限します。

> allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";deny (modify) user=anybody

表D.8 certServer.log.configuration.fileName ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログインスタンスの **fileName** パラメーターの値を表示します。	許可	管理者 エージェント 監査者
modify	ログインスタンスの **fileName** パラメーターの値を表示します。	却下	全ユーザー

D.2.8. certServer.log.content.system

インスタンスのログを表示できるユーザーを制御します。

> allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors"

表D.9 certServer.log.content.system ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログの内容を表示します。すべてのログを一覧表示します。	許可	管理者 エージェント 監査者

D.2.9. certServer.log.content.signedAudit

署名付き監査ログにアクセスできるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (read) group="Auditors"
```

表D.10 certServer.log.content.signedAudit ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	ログの内容を表示します。ログを一覧表示します。	許可	監査者

D.2.10. certServer.registry.configuration

プラグインモジュールの登録に使用されるファイルである管理レジストリーへのアクセスを制御します。現在、これは証明書プロファイルプラグインの登録にのみ使用されます。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.11 certServer.registry.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	管理レジストリー、サポートされているポリシー制約、プロファイルプラグインの設定、およびプロファイルプラグインのリストを表示します。	許可	管理者 エージェント 監査者

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	個々のプロファイル実装プラグインを登録します。	許可	管理者

D.3. 証明書マネージャー固有の ACL

本セクションでは、Certificate Manager 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。CA ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれています。

CA の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、証明書の一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.3.1. certServer.admin.ocsp

Certificate Manager の OCSP 設定へのアクセスを、エンタープライズ OCSP 管理者グループのメンバーに制限します。

```
allow (modify,read) group="Enterprise OCSP Administrators"
```

表D.12 certServer.admin.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	OCSP 設定、OCSP ストア設定、およびデフォルトの OCSP ストアを変更します。	許可	エンタープライズ OCSP 管理者
read	OCSP 設定を読み取ります。	許可	エンタープライズ OCSP 管理者

D.3.2. certServer.ca.certificate

証明書のインポートや取り消しなど、エージェントサービスインターフェイスでの証明書の基本的な管理操作を制御します。デフォルト設定は以下のようになります。

```
allow (import,unrevoke,revoke,read) group="Certificate Manager Agents"
```

表D.13 certServer.ca.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
import	シリアル番号で証明書を取得します。	許可	Certificate Manager Agent
unrevoke	証明書のステータスを失効から変更します。	許可	Certificate Manager Agent
revoke	証明書のステータスを失効に変更します。	許可	Certificate Manager Agent
read	リクエスト ID に基づいて証明書を取得し、リクエスト ID またはシリアル番号に基づいて証明書の詳細を表示します。	許可	Certificate Manager Agent

D.3.3. certServer.ca.certificates

エージェントサービスインターフェイスを介して証明書の一覧表示または取り消しを行う操作を制御します。デフォルト設定は以下のようになります。

```
allow (revoke,list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

表D.14 certServer.ca.certificates ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
revoke	証明書を取り消すか、または証明書失効リスト要求を承認します。TPS から証明書を取り消します。失効要求に関する追加データのプロンプトを表示します。	許可	Certificate Manager Agent 登録マネージャーエージェント
list	検索に基づいて証明書を一覧表示します。シリアル番号の範囲に基づいて証明書の範囲の詳細を取得します。	許可	Certificate Manager Agent 登録マネージャーエージェント

D.3.4. certServer.ca.configuration

Certificate Manager の一般的な設定での操作を制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.15 certServer.ca.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	CRL プラグイン情報、一般的な CA 設定、CA コネクター設定、CRL 発行ポイント設定、CRL プロファイル設定、要求通知設定、失効通知設定、キュー内要求通知設定、および CRL 拡張設定を表示します。CRL 拡張設定および CRL 発行ポイント設定を一覧表示します。	許可	管理者 エージェント 監査者
modify	CRL 発行ポイントを追加し、削除します。一般的な CA 設定、CA コネクター設定、CRL 発行ポイント設定、CRL 設定、要求通知設定、失効通知設定、キュー内要求通知設定、および CRL 拡張設定を変更します。	許可	管理者

D.3.5. certServer.ca.connector

特別なコネクターを CA に送信する操作を制御します。デフォルト設定は以下のようになります。

```
allow (submit) group="Trusted Managers"
```

表D.16 certServer.ca.connector ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	リモート信頼できるマネージャーからのリクエストを送信します。	許可	信頼できるマネージャー

D.3.6. certServer.ca.connectorInfo

コネクター情報へのアクセスを制御して、CA と KRA と間の信頼できる関係を管理します。これらの信頼関係は、CA と KRA が自動的に接続して、主要なアーカイブおよび復元操作を実行できるようにする特別な設定です。これらの信頼関係は、特別なコネクタープラグインを使用して設定されます。

```
allow (read) group="Enterprise KRA Administrators";allow (modify) group="Enterprise KRA Administrators" || group="Subsystem Group"
```

表D.17 certServer.ca.connectorInfo ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	コネクタープラグイン設定を読み取ります。	許可	エンタープライズ KRA 管理者
modify	コネクタープラグイン設定を変更します。	許可	エンタープライズ KRA 管理者 サブシステムグループ

D.3.7. certServer.ca.crl

エージェントサービスインターフェイスを介して CRL の読み取りまたは更新へのアクセスを制御します。デフォルト設定は次のとおりです。

```
allow (read,update) group="Certificate Manager Agents"
```

表D.18 certServer.ca.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	CRL を表示し、CA CRL 処理に関する詳細情報を取得します。	許可	Certificate Manager Agent
update	CRL を更新します。	許可	Certificate Manager Agent

D.3.8. certServer.ca.directory

証明書および CRL の公開に使用される LDAP ディレクトリーへのアクセスを制御します。

```
allow (update) group="Certificate Manager Agents"
```

表D.19 certServer.ca.directory ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
update	CA 証明書、CRL、およびユーザー証明書を LDAP ディレクトリーに公開します。	許可	Certificate Manager Agent

D.3.9. certServer.ca.group

Certificate Manager インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.20 certServer.ca.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。属性内でユーザー証明書の追加または変更	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.3.10. certServer.ca.ocsp

エージェントサービスインターフェイスを介して、使用統計などの OCSP 情報にアクセスして読み取る機能を制御します。

```
allow (read) group="Certificate Manager Agents"
```

表D.21 certServer.ca.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP 使用状況の統計を取得します。	許可	Certificate Manager Agent

D.3.11. certServer.ca.profile

エージェントサービスページで証明書プロファイル設定へのアクセスを制御します。

```
allow (read,approve) group="Certificate Manager Agents"
```

表D.22 certServer.ca.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書プロファイルの詳細を表示します。	許可	Certificate Manager Agent
approve	証明書プロファイルを承認し、有効にします。	許可	Certificate Manager Agent

D.3.12. certServer.ca.profiles

エージェントサービスインターフェイスで証明書プロファイルを一覧表示するアクセスを制御します。

```
allow (list) group="Certificate Manager Agents"
```

表D.23 certServer.ca.profiles ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	証明書プロファイルの一覧表示。	許可	Certificate Manager Agent

D.3.13. certServer.ca.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.24 certServer.ca.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新しいエージェントを登録します。	許可	エンタープライズ管理者
read	既存のエージェント情報を読み取ります。	許可	エンタープライズ管理者

D.3.14. certServer.ca.request.enrollment

登録リクエストの処理方法および割り当て方法を制御します。デフォルト設定は次のとおりです。

```
allow (submit) user="anybody";allow (read,execute,assign,unassign) group="Certificate Manager Agents"
```

表D.25 certServer.ca.request.enrollment ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	登録リクエストを表示します。	許可	Certificate Manager Agent
execute	リクエストの承認状態を変更します。	許可	Certificate Manager Agent
submit	リクエストを送信します。	許可	Anybody
assign	Certificate Manager エージェントに要求を割り当てます。	許可	Certificate Manager Agent
unassign	要求の割り当てを変更します。	許可	Certificate Manager Agent

D.3.15. certServer.ca.request.profile

証明書プロファイルベースの要求の処理を制御します。デフォルト設定は次のとおりです。

```
allow (approve,read) group="Certificate Manager Agents"
```

表D.26 certServer.ca.request.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
approve	証明書プロファイルベースの証明書要求の承認状態を変更します。	許可	Certificate Manager Agent
read	証明書プロファイルベースの証明書要求を表示します。	許可	Certificate Manager Agent

D.3.16. certServer.ca.requests

エージェントサービスインターフェイスで証明書要求を一覧表示できるユーザーを制御します。

```
allow (list) group="Certificate Manager Agents"|| group="Registration Manager Agents"
```

-

表D.27 certServer.ca.requests ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	要求の範囲の詳細を取得し、複雑なフィルターを使用して証明書を検索します。	許可	Certificate Manager Agent 登録マネージャーエージェント

D.3.17. certServer.ca.systemstatus

Certificate Manager インスタンスの統計を表示できるユーザーを制御します。

```
allow (read) group="Certificate Manager Agents"
```

表D.28 certServer.ca.systemstatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	統計を表示します。	許可	Certificate Manager Agent

D.3.18. certServer.ee.certchain

エンドエンティティーの CA 証明書チェーンにアクセスできるユーザーを制御します。

```
allow (download,read) user="anybody"
```

表D.29 certServer.ee.certchain ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
download	CA の証明書チェーンをダウンロードします。	許可	全ユーザー
read	CA の証明書チェーンを表示します。	許可	全ユーザー

D.3.19. certServer.ee.certificate

エンドエンティティーページを介して、証明書のインポートや取り消しなどのほとんどの操作で、証明書にアクセスできるユーザーを制御します。

```
allow (renew,revoke,read,import) user="anybody"
```

表D.30 certServer.ee.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
renew	既存の証明書を更新する要求を送信します。	許可	全ユーザー
revoke	ユーザー証明書の失効要求を送信します。	許可	全ユーザー
read	証明書のシリアル番号または要求 ID に基づいて証明書を取得して表示します。	許可	全ユーザー
import	シリアル番号に基づいて証明書をインポートします。	許可	全ユーザー

D.3.20. certServer.ee.certificates

失効した証明書を一覧表示したり、エンドエンティティーに失効リクエストを送信できるユーザーを制御します。

```
allow (revoke,list) user="anybody"
```

表D.31 certServer.ee.certificates ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
revoke	取り消しする証明書の一覧を送信します。	許可	取り消される証明書の対象は、CA に認証するために提示された証明書と一致させる必要があります。
list	指定の基準に一致する証明書を検索します。	許可	全ユーザー

D.3.21. certServer.ee.crl

エンドエンティティーページから CRL へのアクセスを制御します。

```
allow (read,add) user="anybody"
```

表D.32 certServer.ee.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書失効リストを取得および表示します。	許可	全ユーザー
add	CRL を OCSP サーバーに追加します。	許可	全ユーザー

D.3.22. certServer.ee.profile

プロファイルの詳細を表示したり、プロファイルを介してリクエストを送信したりできるユーザーなど、エンドエンティティーページの証明書プロファイルへのアクセスを制御します。

```
allow (submit,read) user="anybody"
```

表D.33 certServer.ee.profile ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	証明書プロファイルを使用して証明書要求を送信します。	許可	全ユーザー
read	証明書プロファイルの詳細の表示	許可	全ユーザー

D.3.23. certServer.ee.profiles

エンドエンティティーページでアクティブな証明書プロファイルを一覧表示できるユーザーを制御します。

```
allow (list) user="anybody"
```

表D.34 certServer.ee.profiles ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	証明書プロファイルの一覧表示。	許可	全ユーザー

D.3.24. certServer.ee.request.ocsp

クライアントが OCSP 要求を送信する IP アドレスに基づいてアクセスを制御します。

```
allow (submit) ipaddress=".*"
```

表D.35 certServer.ee.request.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	OCSP 要求を送信します。	許可	すべての IP アドレス

D.3.25. certServer.ee.request.revocation

エンドエンティティーページで証明書失効リスト要求を送信できるユーザーを制御します。

```
allow (submit) user="anybody"
```

表D.36 certServer.ee.request.revocation ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	証明書を取り消す要求を送信します。	許可	全ユーザー

D.3.26. certServer.ee.requestStatus

エンドエンティティーページで証明書要求のステータスを表示できるユーザーを制御します。

```
allow (read) user="anybody"
```

表D.37 certServer.ee.requestStatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	その要求に対して発行された証明書の要求およびシリアル番号を取得します。	許可	全ユーザー

D.3.27. certServer.job.configuration

Certificate Manager にジョブを設定できるユーザーを制御します。

```
allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"
```

表D.38 certServer.job.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	基本的なジョブ設定、ジョブインスタンス設定、ジョブプラグイン設定を表示します。ジョブプラグインおよびジョブインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	ジョブプラグインおよびジョブインスタンスを追加および削除します。ジョブプラグインとジョブインスタンスを変更します。	許可	管理者

D.3.28. certServer.profile.configuration

証明書プロファイル設定へのアクセスを制御します。デフォルト設定は次のとおりです。

> allow (read) group="Administrators" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents" || group="Auditors";allow (modify) group="Administrators"

表D.39 certServer.profile.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書プロファイルのデフォルトと制約、入力、出力、入力設定、出力設定、デフォルト設定、ポリシー制約設定、および証明書プロファイルインスタンス設定を表示します。証明書プロファイルプラグインおよび証明書プロファイルインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	証明書プロファイルのデフォルトおよび制約、入力、出力、および証明書プロファイルインスタンスの追加、変更、削除を行います。デフォルトのポリシー制約設定を追加および変更します。	許可	管理者

D.3.29. certServer.publisher.configuration

Certificate Manager の公開設定を表示および編集できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Auditors" || group="Certificate Manager Agents" || group="Registration Manager Agents" || group="Key Recovery Authority Agents" || group="Online Certificate Status Manager Agents";allow (modify) group="Administrators"
```

表D.40 certServer.publisher.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	LDAP サーバーの宛先情報、パブリッシャープラグイン設定、パブリッシャーインスタンス設定、マッパープラグイン設定、マッパーインスタンス設定、ルールプラグイン設定、およびルールインスタンス設定を表示します。パブリッシャープラグインとインスタンス、ルールプラグインとインスタンス、およびマッパープラグインとインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	パブリッシャープラグイン、パブリッシャーインスタンス、マッパープラグイン、マッパーインスタンス、ルールプラグイン、およびルールインスタンスを追加および削除します。パブリッシャーインスタンス、マッパーインスタンス、ルールインスタンス、および LDAP サーバーの宛先情報を変更します。	許可	管理者

D.3.30. certServer.securitydomain.domainxml

ドメインホストの Certificate Manager によってレジストリーに保持されるセキュリティードメイン情報へのアクセスを制御します。セキュリティードメイン設定は、設定中にサブシステムインスタンスによって直接アクセスおよび変更されるため、サブシステムへの適切なアクセスを常に許可する必要があります。そうしないと、設定が失敗する可能性があります。

```
allow (read) user="anybody";allow (modify) group="Subsystem Group"
```

表D.41 certServer.securitydomain.domainxml ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	セキュリティードメイン設定を表示します。	許可	Anybody
modify	インスタンス情報を変更し、インスタンスを追加および削除して、セキュリティードメイン設定を変更します。	許可	サブシステムグループ エンタープライズ管理者

D.4. キーリカバリー認証局固有の ACL

本セクションでは、KRA 向けに特別に適用されるデフォルトのアクセス制御設定を説明します。KRA ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれています。

KRA の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、キーの一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.4.1. certServer.job.configuration

KRA のジョブを設定できるユーザーを制御します。

```
allow (read) group="Administrators" || group="Key Recovery Authority Agents" ||
group="Auditors";allow (modify) group="Administrators"
```

表D.42 certServer.job.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	基本的なジョブ設定、ジョブインスタンス設定、ジョブプラグイン設定を表示します。ジョブプラグインおよびジョブインスタンスを一覧表示します。	許可	管理者 エージェント 監査者
modify	ジョブプラグインおよびジョブインスタンスを追加および削除します。ジョブプラグインとジョブインスタンスを変更します。	許可	管理者

D.4.2. certServer.kra.certificate.transport

KRA のトランスポート証明書を表示できるユーザーを制御します。

```
allow (read) user="anybody"
```

表D.43 certServer.kra.certificate.transport ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	KRA インスタンスのトランスポート証明書を表示します。	許可	全ユーザー

D.4.3. certServer.kra.configuration

KRA の設定を設定および管理するユーザーを制御します。

```
allow (read) group="Administrators" || group="Auditors" || group="Key Recovery Authority Agents" ||
allow (modify) group="Administrators"
```

表D.44 certServer.kra.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	必要なリカバリーエージェントの承認の数を確認します。	許可	管理者 エージェント 監査者
modify	必要なリカバリーエージェントの承認の数を変更します。	許可	管理者

D.4.4. certServer.kra.connector

KRA に接続するために CA に設定された特別なコネクターで要求を送信できるエンティティーを制御します。デフォルト設定は以下のようになります。

```
allow (submit) group="Trusted Managers"
```

表D.45 certServer.kra.connector ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	新しい鍵のアーカイブ要求を送信します (TMS 以外)。	許可	信頼できるマネージャー

D.4.5. certServer.kra.GenerateKeyPair

KRA にキーリカバリー要求を送信できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (execute) group="Key Recovery Authority Agents"
```

表D.46 certServer.kra.GenerateKeyPair ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	サーバー側のキー生成 (TMS のみ) を実行します。	許可	KRA エージェント

D.4.6. certServer.kra.getTransportCert

KRA にキーリカバリー要求を送信できるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (download) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.47 certServer.kra.getTransportCert ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
download	KRA トランスポート証明書を取得します。	許可	エンタープライズ管理者

D.4.7. certServer.kra.group

KRA インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.48 certServer.kra.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.4.8. certServer.kra.key

鍵の表示、リカバリー、またはダウンロードを使用して鍵情報にアクセスできるユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read,recover,download) group="Key Recovery Authority Agents"
```

表D.49 certServer.kra.key ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	重要なアーカイブ記録に関する公開情報を表示します。	許可	KRA エージェント
recover	データベースからキー情報を取得してリカバリー操作を実行します。	許可	KRA エージェント
download	エージェントサービスページからキー情報をダウンロードします。	許可	KRA エージェント

D.4.9. certServer.kra.keys

エージェントサービスページからアーカイブされた鍵を一覧表示できるユーザーを制御します。

```
allow (list) group="Key Recovery Authority Agents"
```

表D.50 certServer.kra.keys ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	アーカイブされた鍵の範囲を検索し、一覧表示します。	許可	KRA エージェント

D.4.10. certServer.kra.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.51 certServer.kra.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新規ユーザーを登録します。	許可	エンタープライズ管理者
read	既存のユーザー情報を読み込みます。	許可	エンタープライズ管理者

D.4.11. certServer.kra.request

エージェントサービスインターフェイスでキーのアーカイブとリカバリー要求を表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.52 certServer.kra.request ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	鍵のアーカイブまたはリカバリー要求を表示します。	許可	KRA エージェント

D.4.12. certServer.kra.request.status

エンドエンティティーページでキーリカバリー要求のステータスを表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.53 certServer.kra.request.status ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	エージェントサービスページでキーリカバリー要求のステータスを取得します。	許可	KRA エージェント

D.4.13. certServer.kra.requests

エージェントサービスインターフェイスでキーのアーカイブとリカバリー要求を一覧表示できるユーザーを制御します。

```
allow (list) group="Key Recovery Authority Agents"
```

表D.54 certServer.kra.requests ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	キーアーカイブおよびリカバリー要求の範囲の詳細を取得します。	許可	KRA エージェント

D.4.14. certServer.kra.systemstatus

KRA インスタンスの統計を表示できるユーザーを制御します。

```
allow (read) group="Key Recovery Authority Agents"
```

表D.55 certServer.kra.systemstatus ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	統計を表示します。	許可	KRA エージェント

D.4.15. certServer.kra.TokenKeyRecovery

トークンのキーリカバリー要求を KRA に送信するユーザーを制御します。これは、失われたトークンを置き換えるための一般的なリクエストです。デフォルト設定は以下のようになります。

```
allow (submit) group="Key Recovery Authority Agents"
```

表D.56 certServer.kra.TokenKeyRecovery ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	トークンリカバリーのキーリカバリー要求を送信または開始します。	許可	KRA エージェント

D.5. オンライン証明書ステータスマネージャー固有の ACL

本セクションでは、Online Certificate Status Manager 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。OCSP レスポンダーの ACL 設定には、「共通 ACL」 に記載の共通 ACL がすべて含まれます。

OCSP の各インターフェイス (管理コンソール、エージェント、およびエンドエンティティーサービスページ) と、CRL の一覧表示やダウンロードなどの一般的な操作に対して、アクセス制御ルールが設定されています。

D.5.1. certServer.ee.crl

エンドエンティティーページから CRL へのアクセスを制御します。

```
allow (read) user="anybody"
```

表D.57 certServer.ee.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	証明書失効リストを取得および表示します。	許可	全ユーザー

D.5.2. certServer.ee.request.ocsp

クライアントが OCSP 要求を送信する IP アドレスに基づいてアクセスを制御します。

```
allow (submit) ipaddress=".*"
```

表D.58 certServer.ee.request.ocsp ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
submit	OCSP 要求を送信します。	許可	すべての IP アドレス

D.5.3. certServer.ocsp.ca

OCSP レスポンダーを指示できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (add) group="Online Certificate Status Manager Agents"
```

表D.59 certServer.ocsp.ca ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
Add	新しい CA に対する OCSP 要求に応答するように OCSP レスポンダーに指示します。	許可	OCSP Manager エージェント

D.5.4. certServer.ocsp.cas

CRL を Online Certificate Status Manager に公開するすべての Certificate Manager を、エージェントサービスインターフェイスで一覧表示できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (list) group="Online Certificate Status Manager Agents"
```

表D.60 certServer.ocsp.cas ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
list	OCSP レスポンダーに CRL を公開する Certificate Manager の一覧を表示します。	許可	エージェント

D.5.5. certServer.ocsp.certificate

証明書のステータスを検証できるユーザーを制御します。デフォルト設定は次のとおりです。

```
allow (validate) group="Online Certificate Status Manager Agents"
```

表D.61 certServer.ocsp.certificate ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
validate	指定の証明書の状態を検証します。	許可	OCSP エージェント

D.5.6. certServer.ocsp.configuration

Certificate Manager の OCSP サービスの設定へのアクセス、表示、または変更が可能なユーザーを制御します。デフォルト設定は以下のようになります。

```
allow (read) group="Administrators" || group="Online Certificate Status Manager Agents" ||
group="Auditors";allow (modify) group="Administrators"
```

表D.62 certServer.ocsp.configuration ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP プラグインの情報、OCSP 設定、および OCSP ストア設定を表示します。OCSP ストアの設定を一覧表示します。	許可	管理者 オンライン証明書ステータスマネージャーエージェント 監査者
modify	OCSP 設定、OCSP ストア設定、およびデフォルトの OCSP ストアを変更します。	許可	管理者

D.5.7. certServer.ocsp.crl

エージェントサービスインターフェイスを介して CRL の読み取りまたは更新へのアクセスを制御します。デフォルト設定は次のとおりです。

```
allow (add) group="Online Certificate Status Manager Agents" || group="Trusted Managers"
```

表D.63 certServer.ocsp.crl ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
add	新しい CRL を、OCSP レスポンダーが管理するものに追加します。	許可	OCSP エージェント 信頼できるマネージャー

D.5.8. certServer.ocsp.group

Online Certificate Status Manager インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.64 certServer.ocsp.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.5.9. certServer.ocsp.info

OCSP レスポンダーに関する情報を読み取ることができるユーザーを制御します。

```
allow (read) group="Online Certificate Status Manager Agents"
```

表D.65 certServer.ocsp.info ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
read	OCSP レスポンダー情報を表示します。	許可	OCSP エージェント

D.6. トークンキーサービス固有の ACL

本セクションでは、トークンキーサービス (TKS) 用に特別に設定されたデフォルトのアクセス制御設定属性を説明します。TKS ACL 設定には、「共通 ACL」 にリストされている共通 ACL がすべて含まれます。

TKS の管理コンソール、およびその他のサブシステムによる TKS へのアクセスに対するアクセス制御ルールを設定できます。

D.6.1. certServer.tks.encrypteddata

データを暗号化できるユーザーを制御します。

```
allow(execute) group="Token Key Service Manager Agents"
```

表D.66 certServer.tks.encrypteddata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	TKS に保存されている暗号化されたデータ。	許可	TKS エージェント

D.6.2. certServer.tks.group

TKS インスタンスのユーザーおよびグループを追加するために内部データベースへのアクセスを制御します。

```
allow (modify,read) group="Administrators"
```

表D.67 certServer.tks.group ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	インスタンスのユーザーおよびグループエントリーを作成、編集、削除します。	許可	管理者
read	インスタンスのユーザーおよびグループエントリーを表示します。	許可	管理者

D.6.3. certServer.tks.importTransportCert

TKS が鍵の送信に使用するトランスポート証明書をインポートするユーザーを制御します。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.68 certServer.tks.importTransportCert ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	トランスポート証明書を更新します。	許可	エンタープライズ管理者
read	トランスポート証明書をインポートします。	許可	エンタープライズ管理者

D.6.4. certServer.tks.keysetdata

TKS が派生して保存する鍵セットに関する情報を表示するユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.69 certServer.tks.keysetdata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	コンマ区切りのキーセットデータを作成します。	許可	TKS エージェント

D.6.5. certServer.tks.registerUser

インスタンス用にエージェントユーザーを作成できるグループまたはユーザーを定義します。デフォルト設定は以下のようになります。

```
allow (modify,read) group="Enterprise CA Administrators" || group="Enterprise KRA Administrators" ||
group="Enterprise OCSP Administrators" || group="Enterprise TKS Administrators" ||
group="Enterprise TPS Administrators"
```

表D.70 certServer.tks.registerUser ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
modify	新しいエージェントを登録します。	許可	エンタープライズ管理者
read	既存のエージェント情報を読み取ります。	許可	エンタープライズ管理者

D.6.6. certServer.tks.sessionkey

TPS に接続するために TKS インスタンスが使用するセッションキーを作成するユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.71 certServer.tks.sessionkey ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	TKS が生成するセッションキーを作成します。	許可	TKS エージェント

D.6.7. certServer.tks.randomdata

ランダムなデータを作成できるユーザーを制御します。

```
allow (execute) group="Token Key Service Manager Agents"
```

表D.72 certServer.tks.randomdata ACL の概要

操作	説明	アクセスの許可または拒否	対象のユーザーまたはグループ
実行	ランダムなデータを生成します。	許可	TKS エージェント

付録E 監査イベント

この付録では、個別の監査イベントとそのパラメーターの説明および形式を提供します。ログ内のすべての監査イベントは、以下の情報を反映しています。

- スレッドの Java 識別子。以下に例を示します。

  ```
  0.localhost-startStop-1
  ```

- イベントが発生したタイムスタンプ。以下に例を示します。

  ```
  [21/Jan/2019:17:53:00 IST]
  ```

- ログソース (14 は SIGNED_AUDIT)。

  ```
  [14]
  ```

- 現在のログレベル (6 はセキュリティー関連のイベント)。『Red Hat Certificate System 計画、インストール、およびデプロイメントのガイド』 の 『ログレベル (メッセージカテゴリー)』 セクションを参照してください。以下に例を示します。

  ```
  [6]
  ```

- ログイベントに関する情報 (ログイベント固有です。特定のログイベントの各フィールドに関する情報は 「監査イベントの説明」 を参照してください。以下に例を示します。

  ```
  [AuditEvent=AUDIT_LOG_STARTUP][SubjectID=$System$][Outcome=Success] audit function startup
  ```

E.1. 監査イベントの説明

以下は、Certificate System で提供される監査イベントの一覧です。

```
########################## SIGNED AUDIT EVENTS ###############################
# Common fields:
# - Outcome: "Success" or "Failure"
# - SubjectID: The UID of the user responsible for the operation
#     "$System$" or "SYSTEM" if system-initiated operation (e.g. log signing).
#
##############################################################################
# Required Audit Events
#
# Event: ACCESS_SESSION_ESTABLISH with [Outcome=Failure]
# Description: This event is used when access session failed to establish.
# Applicable subsystems: CA, KRA, OCSP, TKS, TPS
# Enabled by default: Yes
# Fields:
# - ClientIP: Client IP address.
# - ServerIP: Server IP address.
# - SubjectID: Client certificate subject DN.
# - Outcome: Failure
```

```
    # - Info: Failure reason.
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_ESTABLISH_FAILURE=\
    <type=ACCESS_SESSION_ESTABLISH>:[AuditEvent=ACCESS_SESSION_ESTABLISH]{0}
access session establish failure
    #
    # Event: ACCESS_SESSION_ESTABLISH with [Outcome=Success]
    # Description: This event is used when access session was established successfully.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientIP: Client IP address.
    # - ServerIP: Server IP address.
    # - SubjectID: Client certificate subject DN.
    # - Outcome: Success
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_ESTABLISH_SUCCESS=\
    <type=ACCESS_SESSION_ESTABLISH>:[AuditEvent=ACCESS_SESSION_ESTABLISH]{0}
access session establish success
    #
    # Event: ACCESS_SESSION_TERMINATED
    # Description: This event is used when access session was terminated.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientIP: Client IP address.
    # - ServerIP: Server IP address.
    # - SubjectID: Client certificate subject DN.
    # - Info: The TLS Alert received from NSS
    # - Outcome: Success
    # - Info: The TLS Alert received from NSS
    #
    LOGGING_SIGNED_AUDIT_ACCESS_SESSION_TERMINATED=\
    <type=ACCESS_SESSION_TERMINATED>:[AuditEvent=ACCESS_SESSION_TERMINATED]{0}
access session terminated
    #
    # Event: AUDIT_LOG_SIGNING
    # Description: This event is used when a signature on the audit log is generated (same as "flush"
time).
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: Predefined to be "$System$" because this operation
    #     associates with no user.
    # - Outcome: Success
    # - sig: The base-64 encoded signature of the buffer just flushed.
    #
    LOGGING_SIGNED_AUDIT_AUDIT_LOG_SIGNING_3=[AuditEvent=AUDIT_LOG_SIGNING]
[SubjectID={0}][Outcome={1}] signature of audit buffer just flushed: sig: {2}
    #
    # Event: AUDIT_LOG_STARTUP
    # Description: This event is used at audit function startup.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
```

```
    # - Outcome:
    #
    LOGGING_SIGNED_AUDIT_AUDIT_LOG_STARTUP_2=<type=AUDIT_LOG_STARTUP>:
[AuditEvent=AUDIT_LOG_STARTUP][SubjectID={0}][Outcome={1}] audit function startup
    #
    # Event: AUTH with [Outcome=Failure]
    # Description: This event is used when authentication fails.
    #   In case of TLS-client auth, only webserver env can pick up the TLS violation.
    #   CS authMgr can pick up certificate mismatch, so this event is used.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID:
    # - Outcome: Failure
    #     (obviously, if authentication failed, you won't have a valid SubjectID, so
    #     in this case, SubjectID should be $Unidentified$)
    # - AuthMgr: The authentication manager instance name that did
    #     this authentication.
    # - AttemptedCred: The credential attempted and failed.
    #
    LOGGING_SIGNED_AUDIT_AUTH_FAIL=<type=AUTH>:[AuditEvent=AUTH]{0} authentication
failure
    #
    # Event: AUTH with [Outcome=Success]
    # Description: This event is used when authentication succeeded.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of user who has been authenticated
    # - Outcome: Success
    # - AuthMgr: The authentication manager instance name that did
    #     this authentication.
    #
    LOGGING_SIGNED_AUDIT_AUTH_SUCCESS=<type=AUTH>:[AuditEvent=AUTH]{0}
authentication success
    #
    # Event: AUTHZ with [Outcome=Failure]
    # Description: This event is used when authorization has failed.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of user who has failed to be authorized for an action
    # - Outcome: Failure
    # - aclResource: The ACL resource ID as defined in ACL resource list.
    # - Op: One of the operations as defined with the ACL statement
    #    e.g. "read" for an ACL statement containing "(read,write)".
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_AUTHZ_FAIL=<type=AUTHZ>:[AuditEvent=AUTHZ]{0} authorization
failure
    #
    # Event: AUTHZ with [Outcome=Success]
    # Description: This event is used when authorization is successful.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
```

```
    # - SubjectID: id of user who has been authorized for an action
    # - Outcome: Success
    # - aclResource: The ACL resource ID as defined in ACL resource list.
    # - Op: One of the operations as defined with the ACL statement
    #     e.g. "read" for an ACL statement containing "(read,write)".
    #
    LOGGING_SIGNED_AUDIT_AUTHZ_SUCCESS=<type=AUTHZ>:[AuditEvent=AUTHZ]{0}
authorization success
    #
    # Event: CERT_PROFILE_APPROVAL
    # Description: This event is used when an agent approves/disapproves a certificate profile set by
the
    #   administrator for automatic approval.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of the CA agent who approved the certificate enrollment profile
    # - Outcome:
    # - ProfileID: One of the profiles defined by the administrator
    #     and to be approved by an agent.
    # - Op: "approve" or "disapprove".
    #
    LOGGING_SIGNED_AUDIT_CERT_PROFILE_APPROVAL_4=
<type=CERT_PROFILE_APPROVAL>:[AuditEvent=CERT_PROFILE_APPROVAL][SubjectID={0}]
[Outcome={1}][ProfileID={2}][Op={3}] certificate profile approval
    #
    # Event: CERT_REQUEST_PROCESSED
    # Description: This event is used when certificate request has just been through the approval
process.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of the agent who approves, rejects, or cancels
    #     the certificate request.
    # - Outcome:
    # - ReqID: The request ID.
    # - InfoName: "certificate" (in case of approval), "rejectReason"
    #     (in case of reject), or "cancelReason" (in case of cancel)
    # - InfoValue: The certificate (in case of success), a reject reason in
    #     text, or a cancel reason in text.
    # - CertSerialNum:
    #
    LOGGING_SIGNED_AUDIT_CERT_REQUEST_PROCESSED=
<type=CERT_REQUEST_PROCESSED>:[AuditEvent=CERT_REQUEST_PROCESSED]{0}
certificate request processed
    #
    # Event: CERT_SIGNING_INFO
    # Description: This event indicates which key is used to sign certificates.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome: Success
    # - SKI: Subject Key Identifier of the certificate signing certificate
    # - AuthorityID: (applicable only to lightweight CA)
    #
```

```
    LOGGING_SIGNED_AUDIT_CERT_SIGNING_INFO=<type=CERT_SIGNING_INFO>:
[AuditEvent=CERT_SIGNING_INFO]{0} certificate signing info
    #
    # Event: CERT_STATUS_CHANGE_REQUEST
    # Description: This event is used when a certificate status change request (e.g. revocation)
    #   is made (before approval process).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of uer who performed the action
    # - Outcome:
    # - ReqID: The request ID.
    # - CertSerialNum: The serial number (in hex) of the certificate to be revoked.
    # - RequestType: "revoke", "on-hold", "off-hold"
    #
    LOGGING_SIGNED_AUDIT_CERT_STATUS_CHANGE_REQUEST=
<type=CERT_STATUS_CHANGE_REQUEST>:[AuditEvent=CERT_STATUS_CHANGE_REQUEST]
{0} certificate revocation/unrevocation request made
    #
    # Event: CERT_STATUS_CHANGE_REQUEST_PROCESSED
    # Description: This event is used when certificate status is changed (revoked, expired, on-hold,
    #   off-hold).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of the agent that processed the request.
    # - Outcome:
    # - ReqID: The request ID.
    # - RequestType: "revoke", "on-hold", "off-hold"
    # - Approval: "complete", "rejected", or "canceled"
    #     (note that "complete" means "approved")
    # - CertSerialNum: The serial number (in hex).
    # - RevokeReasonNum: One of the following number:
    #     reason number       reason
    #     ---------------------------------------
    #     0             Unspecified
    #     1             Key compromised
    #     2             CA key compromised (should not be used)
    #     3             Affiliation changed
    #     4             Certificate superceded
    #     5             Cessation of operation
    #     6             Certificate is on-hold
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_CERT_STATUS_CHANGE_REQUEST_PROCESSED=
<type=CERT_STATUS_CHANGE_REQUEST_PROCESSED>:
[AuditEvent=CERT_STATUS_CHANGE_REQUEST_PROCESSED]{0} certificate status change
request processed
    #
    # Event: CLIENT_ACCESS_SESSION_ESTABLISH with [Outcome=Failure]
    # Description: This event is when access session failed to establish when Certificate System acts
as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
```

```
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Failure
    # - Info:
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_ESTABLISH_FAILURE=\
    <type=CLIENT_ACCESS_SESSION_ESTABLISH>:
[AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH]{0} access session failed to establish when
Certificate System acts as client
    #
    # Event: CLIENT_ACCESS_SESSION_ESTABLISH with [Outcome=Success]
    # Description: This event is used when access session was established successfully when
    #   Certificate System acts as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Success
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_ESTABLISH_SUCCESS=\
    <type=CLIENT_ACCESS_SESSION_ESTABLISH>:
[AuditEvent=CLIENT_ACCESS_SESSION_ESTABLISH]{0} access session establish successfully
when Certificate System acts as client
    #
    # Event: CLIENT_ACCESS_SESSION_TERMINATED
    # Description: This event is used when access session was terminated when Certificate System
acts as client.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - ClientHost: Client hostname.
    # - ServerHost: Server hostname.
    # - ServerPort: Server port.
    # - SubjectID: SYSTEM
    # - Outcome: Success
    # - Info: The TLS Alert received from NSS
    #
    LOGGING_SIGNED_AUDIT_CLIENT_ACCESS_SESSION_TERMINATED=\
    <type=CLIENT_ACCESS_SESSION_TERMINATED>:
[AuditEvent=CLIENT_ACCESS_SESSION_TERMINATED]{0} access session terminated when
Certificate System acts as client
    #
    # Event: CMC_REQUEST_RECEIVED
    # Description: This event is used when a CMC request is received.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    #     If CMC requests is signed by an agent, SubjectID should
    #     be that of the agent.
    #     In case of an unsigned request, it would bear $Unidentified$.
    # - Outcome:
```

```
    # - CMCRequest: Base64 encoding of the CMC request received
    #
    LOGGING_SIGNED_AUDIT_CMC_REQUEST_RECEIVED_3=
<type=CMC_REQUEST_RECEIVED>:[AuditEvent=CMC_REQUEST_RECEIVED][SubjectID={0}]
[Outcome={1}][CMCRequest={2}] CMC request received
    #
    # Event: CMC_RESPONSE_SENT
    # Description: This event is used when a CMC response is sent.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    # - Outcome:
    # - CMCResponse: Base64 encoding of the CMC response sent
    #
    LOGGING_SIGNED_AUDIT_CMC_RESPONSE_SENT_3=<type=CMC_RESPONSE_SENT>:
[AuditEvent=CMC_RESPONSE_SENT][SubjectID={0}][Outcome={1}][CMCResponse={2}] CMC
response sent
    #
    # Event: CMC_SIGNED_REQUEST_SIG_VERIFY
    # Description: This event is used when agent signed CMC certificate requests or revocation
requests
    #   are submitted and signature is verified.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: the user who signed the CMC request (success case)
    # - Outcome:
    # - ReqType: The request type (enrollment, or revocation).
    # - CertSubject: The certificate subject name of the certificate request.
    # - SignerInfo: A unique String representation for the signer.
    #
    LOGGING_SIGNED_AUDIT_CMC_SIGNED_REQUEST_SIG_VERIFY=
<type=CMC_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_SIGNED_REQUEST_SIG_VERIFY]{0} agent signed CMC request signature
verification
    #
    # Event: CMC_USER_SIGNED_REQUEST_SIG_VERIFY
    # Description: This event is used when CMC (user-signed or self-signed) certificate requests or
revocation requests
    #   are submitted and signature is verified.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: the user who signed the CMC request (success case)
    # - Outcome:
    # - ReqType: The request type (enrollment, or revocation).
    # - CertSubject: The certificate subject name of the certificate request.
    # - CMCSignerInfo: A unique String representation for the CMC request signer.
    # - info:
    #
    LOGGING_SIGNED_AUDIT_CMC_USER_SIGNED_REQUEST_SIG_VERIFY_FAILURE=
<type=CMC_USER_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_USER_SIGNED_REQUEST_SIG_VERIFY]{0} User signed CMC request
signature verification failure
    LOGGING_SIGNED_AUDIT_CMC_USER_SIGNED_REQUEST_SIG_VERIFY_SUCCESS=
```

```
<type=CMC_USER_SIGNED_REQUEST_SIG_VERIFY>:
[AuditEvent=CMC_USER_SIGNED_REQUEST_SIG_VERIFY]{0} User signed CMC request
signature verification success
   #
   # Event: CONFIG_ACL
   # Description: This event is used when configuring ACL information.
   # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #
   LOGGING_SIGNED_AUDIT_CONFIG_ACL_3=<type=CONFIG_ACL>:
[AuditEvent=CONFIG_ACL][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] ACL
configuration parameter(s) change
   #
   # Event: CONFIG_AUTH
   # Description: This event is used when configuring authentication.
   # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #     --- Password MUST NOT be logged ---
   #
   LOGGING_SIGNED_AUDIT_CONFIG_AUTH_3=<type=CONFIG_AUTH>:
[AuditEvent=CONFIG_AUTH][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] authentication
configuration parameter(s) change
   #
   # Event: CONFIG_CERT_PROFILE
   # Description: This event is used when configuring certificate profile
   #   (general settings and certificate profile).
   # Applicable subsystems: CA
   # Enabled by default: Yes
   # Fields:
   # - SubjectID: id of administrator who performed the action
   # - Outcome:
   # - ParamNameValPairs: A name-value pair
   #     (where name and value are separated by the delimiter ;;)
   #     separated by + (if more than one name-value pair) of config params changed.
   #
   LOGGING_SIGNED_AUDIT_CONFIG_CERT_PROFILE_3=<type=CONFIG_CERT_PROFILE>:
[AuditEvent=CONFIG_CERT_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}]
certificate profile configuration parameter(s) change
   #
   # Event: CONFIG_CRL_PROFILE
   # Description: This event is used when configuring CRL profile
   #   (extensions, frequency, CRL format).
   # Applicable subsystems: CA
   # Enabled by default: Yes
```

```
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_CRL_PROFILE_3=<type=CONFIG_CRL_PROFILE>:
[AuditEvent=CONFIG_CRL_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] CRL
profile configuration parameter(s) change
    #
    # Event: CONFIG_DRM
    # Description: This event is used when configuring KRA.
    #   This includes key recovery scheme, change of any secret component.
    # Applicable subsystems: KRA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #     --- secret component (password) MUST NOT be logged ---
    #
    LOGGING_SIGNED_AUDIT_CONFIG_DRM_3=<type=CONFIG_DRM>:
[AuditEvent=CONFIG_DRM][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}] DRM
configuration parameter(s) change
    #
    # Event: CONFIG_OCSP_PROFILE
    # Description: This event is used when configuring OCSP profile
    #   (everything under Online Certificate Status Manager).
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_OCSP_PROFILE_3=<type=CONFIG_OCSP_PROFILE>:
[AuditEvent=CONFIG_OCSP_PROFILE][SubjectID={0}][Outcome={1}][ParamNameValPairs={2}]
OCSP profile configuration parameter(s) change
    #
    # Event: CONFIG_ROLE
    # Description: This event is used when configuring role information.
    #   This includes anything under users/groups, add/remove/edit a role, etc.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
```

```
    LOGGING_SIGNED_AUDIT_CONFIG_ROLE=<type=CONFIG_ROLE>:
[AuditEvent=CONFIG_ROLE]{0} role configuration parameter(s) change
    #
    # Event: CONFIG_SERIAL_NUMBER
    # Description: This event is used when configuring serial number ranges
    #   (when requesting a serial number range when cloning, for example).
    # Applicable subsystems: CA, KRA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_SERIAL_NUMBER_1=
<type=CONFIG_SERIAL_NUMBER>:[AuditEvent=CONFIG_SERIAL_NUMBER][SubjectID={0}]
[Outcome={1}][ParamNameValPairs={2}] serial number range update
    #
    # Event: CONFIG_SIGNED_AUDIT
    # Description: This event is used when configuring signedAudit.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id of administrator who performed the action
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_SIGNED_AUDIT=<type=CONFIG_SIGNED_AUDIT>:
[AuditEvent=CONFIG_SIGNED_AUDIT]{0} signed audit configuration parameter(s) change
    #
    # Event: CONFIG_TRUSTED_PUBLIC_KEY
    # Description: This event is used when:
    #   1. "Manage Certificate" is used to edit the trustness of certificates
    #      and deletion of certificates
    #   2. "Certificate Setup Wizard" is used to import CA certificates into the
    #      certificate database (Although CrossCertificatePairs are stored
    #      within internaldb, audit them as well)
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: ID of administrator who performed this configuration
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_CONFIG_TRUSTED_PUBLIC_KEY=
<type=CONFIG_TRUSTED_PUBLIC_KEY>:[AuditEvent=CONFIG_TRUSTED_PUBLIC_KEY]{0}
certificate database configuration
    #
    # Event: CRL_SIGNING_INFO
    # Description: This event indicates which key is used to sign CRLs.
    # Applicable subsystems: CA
```

```
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    # - SKI: Subject Key Identifier of the CRL signing certificate
    #
    LOGGING_SIGNED_AUDIT_CRL_SIGNING_INFO=<type=CRL_SIGNING_INFO>:
[AuditEvent=CRL_SIGNING_INFO]{0} CRL signing info
    #
    # Event: DELTA_CRL_GENERATION
    # Description: This event is used when delta CRL generation is complete.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $Unidentified$
    # - Outcome: "Success" when delta CRL is generated successfully, "Failure" otherwise.
    # - CRLnum: The CRL number that identifies the CRL
    # - Info:
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_DELTA_CRL_GENERATION=<type=DELTA_CRL_GENERATION>:
[AuditEvent=DELTA_CRL_GENERATION]{0} Delta CRL generation
    #
    # Event: FULL_CRL_GENERATION
    # Description: This event is used when full CRL generation is complete.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome: "Success" when full CRL is generated successfully, "Failure" otherwise.
    # - CRLnum: The CRL number that identifies the CRL
    # - Info:
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_FULL_CRL_GENERATION=<type=FULL_CRL_GENERATION>:
[AuditEvent=FULL_CRL_GENERATION]{0} Full CRL generation
    #
    # Event: PROFILE_CERT_REQUEST
    # Description: This event is used when a profile certificate request is made (before approval
process).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: The UID of user that triggered this event.
    #     If CMC enrollment requests signed by an agent, SubjectID should
    #     be that of the agent.
    # - Outcome:
    # - CertSubject: The certificate subject name of the certificate request.
    # - ReqID: The certificate request ID.
    # - ProfileID: One of the certificate profiles defined by the
    #     administrator.
    #
    LOGGING_SIGNED_AUDIT_PROFILE_CERT_REQUEST_5=
<type=PROFILE_CERT_REQUEST>:[AuditEvent=PROFILE_CERT_REQUEST][SubjectID={0}]
[Outcome={1}][ReqID={2}][ProfileID={3}][CertSubject={4}] certificate request made with certificate
profiles
```

```
    #
    # Event: PROOF_OF_POSSESSION
    # Description: This event is used for proof of possession during certificate enrollment processing.
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: id that represents the authenticated user
    # - Outcome:
    # - Info: some information on when/how it occurred
    #
    LOGGING_SIGNED_AUDIT_PROOF_OF_POSSESSION_3=
<type=PROOF_OF_POSSESSION>:[AuditEvent=PROOF_OF_POSSESSION][SubjectID={0}]
[Outcome={1}][Info={2}] proof of possession
    #
    # Event: OCSP_ADD_CA_REQUEST_PROCESSED
    # Description: This event is used when an add CA request to the OCSP Responder is processed.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: "Success" when CA is added successfully, "Failure" otherwise.
    # - CASubjectDN: The subject DN of the leaf CA cert in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_ADD_CA_REQUEST_PROCESSED=
<type=OCSP_ADD_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_ADD_CA_REQUEST_PROCESSED]{0} Add CA for OCSP Responder
    #
    # Event: OCSP_GENERATION
    # Description: This event is used when an OCSP response generated is complete.
    # Applicable subsystems: CA, OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $NonRoleUser$
    # - Outcome: "Success" when OCSP response is generated successfully, "Failure" otherwise.
    # - FailureReason:
    #
    LOGGING_SIGNED_AUDIT_OCSP_GENERATION=<type=OCSP_GENERATION>:
[AuditEvent=OCSP_GENERATION]{0} OCSP response generation
    #
    # Event: OCSP_REMOVE_CA_REQUEST_PROCESSED with [Outcome=Failure]
    # Description: This event is used when a remove CA request to the OCSP Responder is processed
and failed.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: Failure
    # - CASubjectDN: The subject DN of the leaf CA certificate in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_REMOVE_CA_REQUEST_PROCESSED_FAILURE=
<type=OCSP_REMOVE_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_REMOVE_CA_REQUEST_PROCESSED]{0} Remove CA for OCSP Responder
has failed
    #
    # Event: OCSP_REMOVE_CA_REQUEST_PROCESSED with [Outcome=Success]
    # Description: This event is used when a remove CA request to the OCSP Responder is processed
```

```
successfully.
    # Applicable subsystems: OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: OCSP administrator user id
    # - Outcome: "Success" when CA is removed successfully, "Failure" otherwise.
    # - CASubjectDN: The subject DN of the leaf CA certificate in the chain.
    #
    LOGGING_SIGNED_AUDIT_OCSP_REMOVE_CA_REQUEST_PROCESSED_SUCCESS=
<type=OCSP_REMOVE_CA_REQUEST_PROCESSED>:
[AuditEvent=OCSP_REMOVE_CA_REQUEST_PROCESSED]{0} Remove CA for OCSP Responder
is successful
    #
    # Event: OCSP_SIGNING_INFO
    # Description: This event indicates which key is used to sign OCSP responses.
    # Applicable subsystems: CA, OCSP
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    # - SKI: Subject Key Identifier of the OCSP signing certificate
    # - AuthorityID: (applicable only to lightweight CA)
    #
    LOGGING_SIGNED_AUDIT_OCSP_SIGNING_INFO=<type=OCSP_SIGNING_INFO>:
[AuditEvent=OCSP_SIGNING_INFO]{0} OCSP signing info
    #
    # Event: ROLE_ASSUME
    # Description: This event is used when a user assumes a role.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID:
    # - Outcome:
    # - Role: One of the valid roles:
    #     "Administrators", "Certificate Manager Agents", or "Auditors".
    #     Note that customized role names can be used once configured.
    #
    LOGGING_SIGNED_AUDIT_ROLE_ASSUME=<type=ROLE_ASSUME>:
[AuditEvent=ROLE_ASSUME]{0} assume privileged role
    #
    # Event: SECURITY_DOMAIN_UPDATE
    # Description: This event is used when updating contents of security domain
    #   (add/remove a subsystem).
    # Applicable subsystems: CA
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: CA administrator user ID
    # - Outcome:
    # - ParamNameValPairs: A name-value pair
    #     (where name and value are separated by the delimiter ;;)
    #     separated by + (if more than one name-value pair) of config params changed.
    #
    LOGGING_SIGNED_AUDIT_SECURITY_DOMAIN_UPDATE_1=
<type=SECURITY_DOMAIN_UPDATE>:[AuditEvent=SECURITY_DOMAIN_UPDATE][SubjectID=
{0}][Outcome={1}][ParamNameValPairs={2}] security domain update
    #
```

```
    # Event: SELFTESTS_EXECUTION
    # Description: This event is used when self tests are run.
    # Applicable subsystems: CA, KRA, OCSP, TKS, TPS
    # Enabled by default: Yes
    # Fields:
    # - SubjectID: $System$
    # - Outcome:
    #
    LOGGING_SIGNED_AUDIT_SELFTESTS_EXECUTION_2=<type=SELFTESTS_EXECUTION>:
[AuditEvent=SELFTESTS_EXECUTION][SubjectID={0}][Outcome={1}] self tests execution (see
selftests.log for details)
```

用語集

A

agent (エージェント)

Certificate System マネージャーで エージェントサービス (agent services) の管理を許可されたグループに属するユーザー。Certificate Manager エージェント、キーリカバリー認証局エージェント (Key Recovery Authority agent)も併せて参照してください。

APDU

アプリケーションプロトコルデータユニット。スマートカードとスマートカードリーダーとの間の通信に使用される通信ユニット (バイトに類似)。

アクセス制御 (access control)

特定ユーザーが実行できるものを制御するプロセス。たとえば、サーバーへのアクセス制御は通常、パスワードまたは証明書によって確立された ID と、そのエンティティーが実行できることに関するルールに基づいています。アクセス制御リスト (ACL)も併せて参照してください。

アクセス制御リスト (ACL)

サーバーが特定のリソースへのアクセス要求を受け取ったときに評価されるアクセスルールの階層を定義するアクセス制御エントリーのコレクション。アクセス制御手順 (access control instructions, ACI)を参照してください。

アクセス制御手順 (access control instructions, ACI)

アクセスを要求するサブジェクトを識別する方法、または特定のサブジェクトに対して許可または拒否される権限を指定するアクセスルール。アクセス制御リスト (ACL)を参照してください。

エージェントサービス (agent services)

1.エージェントに必要な特権が割り当てられた Certificate System サブシステムが提供する HTML ページを通じて、Certificate System agent (エージェント) が管理できるサービス。

2.このようなサービスを管理するための HTML ページ。

エージェント承認登録の設定 (agent-approved enrollment)

証明書の発行前に、エージェントによる要求を承認するのに必要な登録。

属性値アサーション (attribute value assertion, AVA)

attribute = valueの形式のアサーションで、**attribute** は **o** (組織) または **uid** (ユーザー ID) などのタグ、**value** は Red Hat, Inc.やログイン名などの値です。AVA は、証明書の サブジェクト名 (subject name) と呼ばれる、証明書の対象を識別する 識別名 (distinguished name, DN) を形成するために使用されます。

監査ログ (autdit log)

さまざまなシステムイベントを記録するログこのログは署名して、改ざんされなかった証明を提供でき、auditor ユーザーのみが読み取りできます。

監査者 (auditor)

署名付き監査ログを表示できる特権ユーザー。

管理者 (administrator)

1 つ以上の Certificate System マネージャーをインストールおよび設定し、それらの特権ユーザーまたはエージェントをセットアップする人。agent (エージェント)も併せて参照してください。

自動登録 (automated enrollment)

人の介入なしに、エンドエンティティー登録の自動認証を可能にする Certificate System サブシステムを設定する方法。この形式の認証では、認証モジュールの処理を正常に完了した証明書要求が、プロファイル処理と証明書の発行に対して自動的に承認されます。

認可 (authorization)

サーバーが制御するリソースにアクセスするパーミッション。承認は通常、リソースに関連付けられた ACL がサーバーによって評価された後に行われます。アクセス制御リスト (ACL)を参照してください。

認証 (authentication)

自信を持って識別すること。コンピューター化された送信の当事者が偽者ではないことを保証すること。認証には通常、パスワード、証明書、PIN、またはその他の情報を使用して、コンピューターネットワーク上で ID を検証することが含まれます。パスワードベースの認証 (password-based authentication)、証明書ベースの認証 (certificate-based authentication)、クライアント認証 (client authentication)、サーバー認証 (server authentication) も併せて参照してください。

認証モジュール (authentication module)

ルールのセット (として実装されますJava™クラス) エンドエンティティー、エージェント、管理者、または Certificate System サブシステムと対話する必要があるその他のエンティティーを認証するため。通常のエンドユーザー登録の場合は、ユーザーが登録フォームで要求された情報を入力した後、登録サーブレットはそのフォームに関連付けられた認証モジュールを使用して情報を検証し、ユーザーの ID を認証します。サーブレット (servlet)を参照してください。

高度暗号化標準 (Advanced Encryption Standard, AES)

Advanced Encryption Standard (AES) は、その前身の Data Encryption Standard (DES) と同様に、FIPS 承認の対称鍵暗号化標準です。AES は 2002 年に米国政府によって採用されました。AES-128、AES-192、および AES-256 の 3 つのブロック暗号を定義します。NIST (National Institute of Standards and Technology) は、米国で AES 標準を定義しています。FIPS PUB 197。詳細は、http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf を参照してください。

B

バインド DN (bind DN)

Red Hat Directory Server への認証にパスワードとともに使用される識別名 (DN) の形式のユーザー ID。

C

CA サーバーキー (CA server key)

CA サービスを提供するサーバーの SSL サーバーキー。

CA 署名鍵 (CA signing key)

CA 証明書の公開鍵に対応する秘密鍵。CA は、その署名キーを使用して証明書および CRL に署名します。

CA 証明書 (CA certificate)

認証局を識別する証明書。認証局 (certificate authority, CA)、subordinate CA、ルート CAも併せて参照してください。

CA 階層 (CA hierarchy)

ルート CA が下位 CA に証明書を発行する権限を委任する CA の階層。下位 CA は、発行ステータスを他の CA に委譲して階層を拡張することもできます。認証局 (certificate authority, CA)、subordinate CA、ルート CAも併せて参照してください。

Certificate Manager

認証局として機能する独立した Certificate System サブシステム。Certificate Manager インスタンスは、証明書を発行、更新、および取り消します。証明書は、CRL とともに LDAP ディレクトリーに公開できます。エンドエンティティーからのリクエストを受け入れます。認証局 (certificate authority, CA)を参照してください。

Certificate Manager エージェント

Certificate Manager のエージェントサービスの管理が許可されているグループに所属するユーザーです。これらのサービスには、証明書要求にアクセスして変更 (承認および拒否) し、証明書を発行する機能が含まれています。

Certificate Request Message Format (CRMF)

X.509 証明書の管理に関連するメッセージに使用される形式。この形式は CMMF のサブセットです。証明書管理メッセージ形式 (Certificate Management Message Formats, CMMF) も併せて参照してください。詳細は、https://tools.ietf.org/html/rfc2511 を参照してください。

Certificate System

Red Hat Certificate System、暗号化メッセージ構文 (Cryptographic Message Syntax, CS) を参照してください。

Certificate System コンソール

1 つの Certificate System インスタンスで開くことができるコンソール。Certificate System コンソールを使用すると、Certificate System 管理者は、対応する Certificate System インスタンスの設定設定を制御できます。

Certificate System サブシステム

5 つの Certificate System マネージャー (Certificate Manager、Online Certificate Status Manager、キーリカバリー認証局、Token Key Service、または Token Processing System) の 1 つ。

CMC

暗号化メッセージ構文 (CMC) を介した証明書管理メッセージを参照してください。

CMC 登録 (CMC Enrollment)

署名された登録または署名された失効要求のいずれかを、エージェントの署名証明書を使用して Certificate Manager に送信できるようにする機能。これらの要求は Certificate Manager によって自動的に処理されます。

CMMF

証明書管理メッセージ形式 (Certificate Management Message Formats, CMMF) を参照してください。

CRL

証明書失効リスト (certificate revocation list, CRL) を参照してください。

CRMF

Certificate Request Message Format (CRMF) を参照してください。

CSP

暗号化サービスプロバイダー (cryptographic service provider, CSP) を参照してください。

クライアント SSL 証明書 (client SSL certificate)

SSL プロトコルを使用してサーバーにクライアントを識別するために使用する証明書。セキュアソケットレイヤー (Secure Sockets Layer, SSL)を参照してください。

クライアント認証 (client authentication)

名前とパスワード、または証明書とデジタル署名されたデータなどを使用して、サーバーに対してクライアントを識別するプロセス。証明書ベースの認証 (certificate-based authentication)、パスワードベースの認証 (password-based authentication)、サーバー認証 (server authentication) を参照してください。

チェーン認証局 (chained CA)

リンクされた CA (linked CA) を参照してください。

ペア間の証明書 (cross-pair certificate)

ある CA から別の CA に発行され、信頼の輪を形成するために両方の CA によって保存される証明書。2 つの CA は相互に証明書を発行し、両方のクロスペア証明書を証明書ペアとして格納します。

信頼チェーン (chain of trust)

証明書チェーン (certificate chain) を参照してください。

暗号アルゴリズム (cryptographic algorithm)

encryption や 復号化 (decryption) などの暗号化操作を実行するために使用される一連のルールまたは指示。

暗号モジュール (cryptographic module)

PKCS #11 モジュールを参照してください。

暗号化 (cipher)

暗号アルゴリズム (cryptographic algorithm) を参照してください。

暗号化サービスプロバイダー (cryptographic service provider, CSP)

PKCS #11 で定義されているような標準インターフェイスを使用してそのようなサービスを要求するソフトウェアに代わって、キー生成、キーストレージ、暗号化などの暗号化サービスを実行する暗号化モジュール。

暗号化メッセージ構文 (CMC) を介した証明書管理メッセージ

Certificate Manager への証明書の要求を伝えるために使用するメッセージ形式。Internet Engineering Task Force (IETF) PKIX の作業グループから提案された標準。詳細は、https://tools.ietf.org/html/draft-ietf-pkix-cmc-02 を参照してください。

暗号化メッセージ構文 (Cryptographic Message Syntax, CS)

CMMF などの任意のメッセージにデジタル署名、ダイジェスト、認証、または暗号化するために使用される構文。

相互認証 (cross-certification)

異なる証明書階層またはチェーン内の 2 つの CA による証明書交換クロス認定は、両方の階層に対応できるように信頼チェーンを拡張します。認証局 (certificate authority, CA) も併せて参照してください。

証明書 (certificate)

X.509 標準に準拠してフォーマットされたデジタルデータで、個人、会社などのエンティティー名 (証明書の サブジェクト名 (subject name)) を指定し、証明書に含まれる 公開鍵 (public key) もそのエンティティーに属することを証明するもの。証明書は 認証局 (certificate authority, CA) により発行され、デジタル署名されます。証明書の有効性は、公開鍵暗号 (public-key cryptography) の手法で CA の デジタル署名 (digital signature) を確認して検証できます。公開鍵インフラストラクチャー (public-key infrastructure, PKI) で信頼されるために、証明書は、PKI に登録されているその他のエンティティーによって信頼されている CA により発行および署名される必要があります。

証明書のフィンガープリント (certificate fingerprint)

証明書に関連付けられた 一方向ハッシュ (one-way hash)。番号は証明書自体の一部ではありませんが、証明書の内容にハッシュ関数を適用することによって生成されます。証明書の内容が 1 文字でも変更されると、同じ関数で異なる番号が生成されます。したがって、証明書のフィンガープリントを使用して、証明書が改ざんされていないことを確認できます。

証明書チェーン (certificate chain)

連続した認証局によって署名された証明書の階層セット。CA 証明書は 認証局 (certificate authority, CA) を識別し、その認証局が発行した証明書の署名に使用されます。CA 証明書は、親 CA の CA 証明書により署名され、ルート CA まで署名できます。Certificate System により、エンドエンティティーが証明書チェーンのすべての証明書を取得できるようになります。

証明書プロファイル (certificate profile)

特定のタイプの登録を定義する一連の設定設定。証明書プロファイルは、証明書プロファイルの認証方法とともに、特定のタイプの登録のポリシーを設定します。

証明書ベースの認証 (certificate-based authentication)

証明書および公開鍵暗号に基づく認証。パスワードベースの認証 (password-based authentication) も併せて参照してください。

証明書失効リスト (certificate revocation list, CRL)

X.509 標準で定義されているように、認証局 (certificate authority, CA) により生成され署名された、失効した証明書のシリアル番号ごとのリスト。

証明書拡張 (certificate extensions)

X.509 v3 証明書には、証明書に任意の数の追加フィールドを追加できる拡張フィールドが含まれています。証明書拡張は、代替サブジェクト名や使用制限などの情報を証明書に追加する方法を提供します。PKIX ワーキンググループによって、いくつかの標準拡張機能が定義されています。

証明書管理メッセージ形式 (Certificate Management Message Formats, CMMF)

エンドエンティティーから Certificate Manager に証明書要求と失効要求を伝達し、エンドエンティティーにさまざまな情報を送信するために使用されるメッセージ形式。Internet Engineering Task Force (IETF) PKIX の作業グループから提案された標準。CMMF は、別の提案された標準 (暗号化メッセージ構文 (CMC) を介した証明書管理メッセージ) に含まれています。詳細は、https://tools.ietf.org/html/draft-ietf-pkix-cmmf-02 を参照してください。

認証局 (certificate authority, CA)

証明書が特定を意図している個人またはエンティティーの ID を検証した後、証明書 (certificate) を発行する信頼されたエンティティー。CA は証明書を更新し、取り消し、CRL を生成します。証明書の発行者フィールドで指定されたエンティティーは、常に CA です。認証局は、独立したサードパーティー、または Red Hat Certificate System などの証明書発行サーバーソフトウェアを使用する個人または組織にすることができます。

D

キーリカバリー認証局

エンドエンティティーの RSA 暗号化キーの長期アーカイブとリカバリーを管理する任意の独立した Certificate System サブシステム。Certificate Manager は、新しい証明書を発行する前に、キーリカバリー機関を使用してエンドエンティティーの暗号化キーをアーカイブするように設定できます。キーリカバリー機関は、エンドエンティティーが機密性の高い電子メールなど、組織がいつかリカバリーする必要のあるデータを暗号化している場合にのみ役立ちます。デュアルキーペアをサポートするエンドエンティティーでのみ使用できます。2 つの個別のキーペアで、1 つは暗号化用、もう 1 つはデジタル署名用です。

キーリカバリー認証局のストレージキー (Key Recovery Authority storage key)

キーリカバリー機関の秘密トランスポートキーで復号された後、エンドエンティティーの暗号化キーを暗号化するためにキーリカバリー機関によって使用される特別なキー。ストレージキーがキーリカバリー機関を離れることはありません。

キーリカバリー認証局のトランスポート証明書 (Key Recovery Authority transport certificate)

エンドエンティティーがキーリカバリー機関に転送するためにエンティティーの暗号化キーを暗号化するために使用する公開キーを認証します。キーリカバリー機関は、認証された公開鍵に対応する秘密鍵を使用して、ストレージ鍵で暗号化する前に、エンドエンティティーの鍵を復号します。

キーリカバリー認証局エージェント (Key Recovery Authority agent)

要求キューの管理や HTML ベースの管理ページを使用したリカバリー操作の許可など、キーリカバリー機関のエージェントサービスの管理を許可されたグループに属するユーザー。

キーリカバリー認証局リカバリーエージェント (Key Recovery Authority recovery agent)

キーリカバリー認証局 のストレージキーの一部を所有している **n 人中 m 人** の 1 人。

デジタル ID (digital ID)

証明書 (certificate)を参照してください。

デジタル署名 (digital signature)

デジタル署名を作成するため、署名ソフトウェアは最初に、新しく発行された証明書など、署名するデータから 一方向ハッシュ (one-way hash) を作成します。次に、一方向ハッシュは署名者の秘密鍵で暗号化されます。作成されるデジタル署名は、署名されるデータごとに一意になります。1 つのコンマがメッセージに追加されていても、そのメッセージのデジタル署名が変更されます。署名者の公開鍵を使用したデジタル署名の復号に成功し、同じデータの別のハッシュと比較することで、改ざんの検出 (tamper detection) が可能になります。公開鍵を含む証明書の 証明書チェーン (certificate chain) の検証により、署名側の認証が提供されます。否認防止 (nonrepudiation)、encryptionも併せて参照してください。

デュアルキーペア (dual key pair)

2 つの個別の証明書に対応する、2 つの公開鍵と秘密鍵のペア (合計 4 つの鍵)。一方のペアの秘密鍵は署名操作に使用され、もう一方のペアの公開鍵と秘密鍵は暗号化および復号操作に使用されます。各ペアは個別の 証明書 (certificate) に対応します。暗号鍵 (encryption key)、公開鍵暗号 (public-key cryptography)、署名鍵も併せて参照してください。

デルタ CRL (delta CRL)

最後の完全な CRL が発行されてから取り消された証明書の一覧を含む CRL。

分散ポイント (distribution points)

証明書セットを定義するのに CRL に使用されます。各ディストリビューションポイントは、発行する証明書のセットにより定義されます。CRL は、特定のディストリビューションポイント用に作成できます。

復号化 (decryption)

暗号化されたデータのスクランブル解除。encryptionを参照してください。

識別名 (distinguished name, DN)

証明書の件名を特定する一連の AVA。属性値アサーション (attribute value assertion, AVA) を参照してください。

E

encryption

その意味を偽装する方法で情報をスクランブルします。復号化 (decryption)を参照してください。

extensions フィールド

証明書拡張 (certificate extensions)を参照してください。

エンドエンティティー (end entity)

公開鍵インフラストラクチャー (public-key infrastructure, PKI) において、人、ルーター、サーバー、または 証明書 (certificate) を使用してそれ自体を特定するエンティティー。

エンロールメント (enrollment)

公開鍵インフラストラクチャー (public-key infrastructure, PKI) で使用する X.509 証明書を要求および受信するプロセス。**登録** としても知られています。

傍受 (eavesdropping)

情報が意図されていないエンティティーによってネットワークを介して送信された情報の不正な傍受。

暗号鍵 (encryption key)

暗号化のみに使用される秘密鍵。暗号化鍵とその同等の公開鍵、および 署名鍵 とその同等の公開鍵は、デュアルキーペア (dual key pair) を構成します。

楕円曲線暗号 (Elliptic Curve Cryptography, ECC)

暗号鍵の基礎となる数学的な問題に対して、楕円曲線を用いて加算対数を作成する暗号アルゴリズム。ECC 暗号は、RSA 暗号よりも効率的に使用でき、本質的に複雑であるため、RSA 暗号よりも小さいビットで強力です。

F

Federal Bridge Certificate Authority (FBCA)

2 つの CA が相互にクロスペア証明書を発行し、2 つのクロスペア証明書を単一の証明書ペアとして格納することにより、信頼の輪を形成する設定。

FIPS PUBS 140

FIPS PUBS (Federal Information Standards Publications) 140 は、データの暗号化と復号、またはデジタル署名の作成や検証などの他の暗号化操作を実行する暗号化モジュール、ハードウェア、またはソフトウェアの実装に関する米国政府の標準です。米国政府に販売される多くの製品は、1 つ以上の FIPS 標準に準拠する必要があります。http://www.nist.gov を参照してください。

ファイアウォール (firewall)

2 つ以上のネットワーク間の境界を強制するシステムまたはシステムの組み合わせ。

フィンガープリント (fingerprint)

証明書のフィンガープリント (certificate fingerprint) を参照してください。

I

IP スプーフィング (IP spoofing)

クライアント IP アドレスの禁止。

なりすまし (impersonation)

ネットワークを介して送信される情報の意図された受信者を装う行為。なりすましには、スプーフィング (spoofing) と 詐称 (misrepresentation) の 2 種類があります。

中間認証局 (intermediate CA)

ルート CA と 証明書チェーン (certificate chain) 内の発行済み証明書との間の証明書を持つ CA。

入力 (input)

証明書プロファイル機能のコンテキストでは、特定の証明書プロファイルの登録フォームを定義します。各入力が設定され、この登録用に設定されたすべての入力から登録フォームが動的に作成されます。

J

JAR ファイル (JAR file)

Java™ アーカイブ (JAR) 形式 (archive (JAR) format) に従って編成されたファイルの圧縮コレクションのデジタルエンベロープ。

Java™ アーカイブ (JAR) 形式 (archive (JAR) format)

デジタル署名、インストーラスクリプト、およびその他の情報をディレクトリー内のファイルに関連付けるための一連の規則。

Java™ セキュリティーサービス (JSS)

あJava™ネットワークセキュリティーサービス (NSS) によって実行されるセキュリティー操作を制御するためのインターフェイス。

Java™ ネイティブインターフェイス (JNI) (Native Interface)

特定のプラットフォーム上の Java™ 仮想マシン (JVM) の異なる実装間でバイナリー互換性を提供する標準的なプログラミングインターフェイスで、単一のプラットフォーム用に C や C++ などの言語で記述された既存のコードを Java™ にバインドできるようにします。http://java.sun.com/products/jdk/1.2/docs/guide/jni/index.html を参照してください。

Java™ 暗号アーキテクチャー (Cryptography Architecture, JCA)

暗号化サービス用に Sun Microsystems によって開発された API 仕様および参照。http://java.sun.com/products/jdk/1.2/docs/guide/security/CryptoSpec.Introduction を参照してください。

Java™ 開発キット (Development Kit, JDK)

Java™ プログラミング言語を使用してアプリケーションとアプレットを開発するために Sun Microsystems が提供するソフトウェア開発キット。

K

KEA

鍵交換アルゴリズム (Key Exchange Algorithm, KEA) を参照してください。

鍵 (key)

データを暗号化または復号化するために、暗号アルゴリズム (cryptographic algorithm) が使用する大きな数値。たとえば、あるユーザーの 公開鍵 (public key) を使用すると、他のユーザーがそのユーザー宛てのメッセージを暗号化できます。その後、メッセージは対応する プライベートキー を使用して復号化する必要があります。

鍵交換 (key exchange)

クライアントとサーバーが SSL セッション時に両方で使用する対称鍵を決定するために実行する手順。

鍵交換アルゴリズム (Key Exchange Algorithm, KEA)

米国政府が鍵交換に使用するアルゴリズム。

L

Lightweight Directory Access Protocol (LDAP)

TCP/IP および複数のプラットフォームで実行されるためのディレクトリーサービスプロトコル。LDAP は、X.500 ディレクトリーへのアクセスに使用される Directory Access Protocol (DAP) の簡易バージョンです。LDAP は IETF の変更制御下にあり、インターネット要件を満たすために進化しています。

リンクされた CA (linked CA)

公開サードパーティーの CA によって署名される証明書が内部でデプロイされる 認証局 (certificate authority, CA)。内部 CA は、発行する証明書のルート CA として機能し、サードパーティー CA は、同じサードパーティールート CA にリンクされている他の CA によって発行された証明書のルート CA として機能します。チェーン CA としても知られており、異なるパブリック CA で使用される他の用語も使用します。

M

MD5

Ronald Rivest によって開発されたメッセージダイジェストアルゴリズム。一方向ハッシュ (one-way hash)も併せて参照してください。

メッセージダイジェスト (message digest)

一方向ハッシュ (one-way hash)を参照してください。

手動認証 (manual authentication)

各証明書要求の人間による承認を必要とする Certificate System サブシステムを設定する方法。この形式の認証では、サーブレットは、認証モジュールの処理が成功した後、証明書要求を要求キューに転送します。次に、適切な権限を持つエージェントは、プロファイルの処理と証明書の発行を続行する前に、各要求を個別に承認する必要があります。

詐称 (misrepresentation)

そうではない個人または組織としてのエンティティーの提示。たとえば、Web サイトが実際にはクレジットカードでの支払いを行いますが、商品を送信しないサイトである場合、その Web サイトは家具店のふりをする可能性があります。詐称は なりすまし (impersonation) の 1 つの形式です。スプーフィング (spoofing)も併せて参照してください。

N

non-TMS

トークン以外の管理システム。スマートカードを直接処理 **しない** サブシステム (CA、および任意で KRA と OCSP) の設定を指します。

トークン管理システム参照

ネットワークセキュリティーサービス (Network Security Services, NSS)

セキュリティー対応の通信アプリケーションのクロスプラットフォーム開発をサポートするように設計されたライブラリーのセット。NSS ライブラリーを使用して構築されたアプリケーションは、認証、改ざん検出、および暗号化のための セキュアソケットレイヤー (Secure Sockets Layer, SSL) プロトコル、および暗号化トークンインターフェイスのための PKCS #11 プロトコルをサポートします。NSS は、ソフトウェア開発キットとして個別に利用できます。

否認防止 (nonrepudiation)

メッセージの送信を拒否するためのメッセージの送信者による信頼性。デジタル署名 (digital signature) は、否認防止手段の 1 つです。

O

OCSP

オンライン証明書ステータスプロトコル

オブジェクトの署名 (object signing)

ソフトウェア開発者が Java コード、JavaScript スクリプト、またはあらゆる種類のファイルに署名し、ユーザーが署名者を識別し、署名されたコードによってローカルシステムリソースへのアクセスを制御できるようにするファイル署名の方法。

オブジェクト署名証明書

関連付けられた秘密鍵がオブジェクトの署名に使用される証明書。オブジェクトの署名 (object signing) に関連しています。

一方向ハッシュ (one-way hash)

1.ハッシュアルゴリズムを使用して任意の長さのデータから生成された固定長の数。メッセージダイジェストとも呼ばれる数字は、ハッシュされたデータに固有のものです。1 文字を削除または変更しても、データの変更は異なります。

2.ハッシュされたデータの内容をハッシュから推測することはできません。

出力 (output)

証明書プロファイル機能のコンテキストでは、特定の証明書プロファイルの証明書登録が成功した結果のフォームを定義します。各出力が設定され、この登録に設定されたすべての出力からフォームを動的に作成します。

操作 (operation)

アクセス制御命令で許可または拒否されている、読み取りや書き込みなどの特定の操作。

P

PKCS #10

証明書要求を制御する公開鍵暗号標準規格。

PKCS #11

スマートカードなどの暗号化トークンを管理する公開鍵暗号化標準。

PKCS #11 モジュール

PKCS #11 インターフェイスを介して、暗号化サービス (暗号化や復号など) を提供する暗号化デバイスのドライバー。**暗号化モジュール** または **暗号化サービスプロバイダー** とも呼ばれる PKCS #11 モジュールは、ハードウェアまたはソフトウェアのいずれかで実装できます。PKCS #11 モジュールには、常に 1 つ以上のスロットがあり、スマートカードなどの物理リーダーの形式で物理ハードウェアスロットとして、またはソフトウェアの概念スロットとして実装できます。PKCS #11 モジュールの各スロットには、トークンを追加できます。このトークンは、暗号化サービスを実際に提供し、必要に応じて証明書と鍵を保存するハードウェアまたはソフトウェアのデバイスです。Red Hat は、Certificate System とともに、ビルトイン PKCS #11 モジュールを提供します。

PKCS #12

鍵の移植性を管理する公開鍵暗号化標準。

PKCS #7

署名と暗号化を制御する公開鍵暗号標準。

POA (proof-of-archival)

キーのシリアル番号、キーリカバリー機関の名前、対応する証明書の サブジェクト名 (subject name)、およびアーカイブの日付など、アーカイブされたエンドエンティティーキーに関する情報を含む秘密鍵リカバリー機関のトランスポートキーで署名されたデータ。署名されたアーカイブ証明データは、キーのアーカイブ操作が成功した後、キーリカバリー機関から Certificate Manager に返される応答です。キーリカバリー認証局のトランスポート証明書 (Key Recovery Authority transport certificate)も併せて参照してください。

パスワードベースの認証 (password-based authentication)

名前とパスワードによる確実な識別。認証 (authentication)、証明書ベースの認証 (certificate-based authentication)も併せて参照してください。

プライベートキー

公開鍵暗号で使用されるキーペアの 1 つ。秘密鍵は秘密を保持し、対応する 公開鍵 (public key) で暗号化されたデータの復号に使用されます。

公開鍵 (public key)

公開鍵暗号で使用されるキーペアの 1 つ。公開鍵は自由に配布され、証明書 (certificate) の一部として公開されます。これは通常、公開鍵の所有者に送信されるデータを暗号化するために使用され、所有者は対応する プライベートキー でデータを復号化します。

公開鍵インフラストラクチャー (public-key infrastructure, PKI)

ネットワーク環境での公開鍵暗号化と X.509 v3 証明書の使用を容易にする標準とサービス。

公開鍵暗号 (public-key cryptography)

エンティティーがその ID を電子的に検証したり、電子データに署名して暗号化したりできるようにする、確立された技術と標準のセット。公開鍵と秘密鍵の 2 つの鍵が関係します。公開鍵 (public key) は、特定の ID にキーを関連付ける証明書の一部として公開されます。対応する秘密鍵はシークレットに保存されます。公開鍵で暗号化したデータは、秘密鍵でのみ復号できます。

R

RC2, RC4

Rivest による RSA データセキュリティー向けに開発された暗号化アルゴリズム。暗号アルゴリズム (cryptographic algorithm)も併せて参照してください。

Red Hat Certificate System

証明書を作成、デプロイ、および管理するための高度に設定可能なソフトウェアコンポーネントとツールのセット。Certificate System は、さまざまな物理的な場所にあるさまざまな Certificate System インスタンスにインストールできる 5 つの主要なサブシステム (Certificate Manager、Online Certificate Status Manager、キーリカバリー認証局、Token Key Service、および Token Processing System) で構成されています。

RSA アルゴリズム (RSA algorithm)

Rivest-Shamir-Adleman の略で、暗号化と認証の両方の公開鍵アルゴリズムです。Ronald Rivest、Adi Shamir、および Leonard Adleman により開発され、1978 で導入されました。

RSA キー交換 (RSA key exchange)

RSA アルゴリズムに基づいた SSL の鍵交換アルゴリズム。

ルート CA

証明書チェーンの最上位にある、自己署名証明書を持つ 認証局 (certificate authority, CA) 。CA 証明書 (CA certificate)、subordinate CAも併せて参照してください。

登録 (registration)

エンロールメント (enrollment)を参照してください。

S

SHA

セキュアなハッシュアルゴリズム (米国政府が使用するハッシュ関数)。

SSL

セキュアソケットレイヤー (Secure Sockets Layer, SSL)を参照してください。

subordinate CA

証明書が別の下位 CA またはルート CA によって署名されている認証局。CA 証明書 (CA certificate)、ルート CAを参照してください。

サブジェクト (subject)

証明書 (certificate) で識別されるエンティティー。特に、証明書のサブジェクトフィールドには、認定されたエンティティーを一意に識別する サブジェクト名 (subject name) が含まれます。

サブジェクト名 (subject name)

証明書 (certificate) の サブジェクト (subject) を一意に表す 識別名 (distinguished name, DN) 。

サンドボックス (sandbox)

Java™ コードが動作する必要がある、注意して定義された制限の Java™ 用語。

サーバー SSL 証明書 (server SSL certificate)

セキュアソケットレイヤー (Secure Sockets Layer, SSL) プロトコルを使用して、クライアントに対してサーバーを識別するために使用する証明書。

サーバー認証 (server authentication)

クライアントにサーバーを特定するプロセス。クライアント認証 (client authentication)も併せて参照してください。

サーブレット (servlet)

Certificate System サブシステムに代わってエンドエンティティーとの特定の種類の相互作用を処理する Java™ コード。たとえば、証明書の登録、失効、およびキーリカバリー要求は、それぞれ別のサーブレットで処理されます。

シングルサインオン (single sign-on)

1.Certificate System において、内部データベースとトークンのパスワードを保管することにより、Red Hat Certificate System へのサインオン方法を簡素化するパスワード。ユーザーがログインするたびに、この単一のパスワードを入力する必要があります。

2.ユーザーが 1 台のコンピューターに一度ログインし、ネットワーク内のさまざまなサーバーによって自動的に認証される機能。部分的なシングルサインオンソリューションは、さまざまなサーバーで使用されるパスワードを自動的に追跡するメカニズムなど、さまざまな形式をとることができます。証明書は、公開鍵インフラストラクチャー (public-key infrastructure, PKI) 内のシングルサインオンをサポートします。ユーザーは、ローカルクライアントの秘密鍵データベースに一度ログインすると、クライアントソフトウェアが動作している限り、証明書ベースの認証 (certificate-based authentication) を使用して、ユーザーがアクセスを許可されている組織内の各サーバーにアクセスできます。

スプーフィング (spoofing)

他人のふりをします。たとえば、あるユーザーがメールアドレス **jdoe@example.com** やコンピューターから、**www.redhat.com** と呼ばれるサイトとして誤って特定できます。スプーフィングは、なりすまし (impersonation) の一形態です。詐称 (misrepresentation)も併せて参照してください。

スマートカード (smart card)

マイクロプロセッサーを搭載し、キーや証明書などの暗号化情報を格納し、暗号化操作を実行する小さなデバイス。スマートカードには、PKCS #11 インターフェイスの一部または全体が実装されています。

スロット (slot)

ハードウェアまたはソフトウェアのいずれかに実装された PKCS #11 モジュール の一部。これにはトークン (token) が含まれています。

セキュアなチャンネル

TPS とスマートカード間のセキュリティーアソシエーション。TKS とスマートカード APDU によって生成された共有マスターキーに基づいて暗号化された通信を可能にします。

セキュアソケットレイヤー (Secure Sockets Layer, SSL)

クライアントとサーバーとの間の相互認証と、認証および暗号化された接続の確立を可能にするプロトコル。SSL は、TCP/IP より上で、HTTP、LDAP、IMAP、NNTP、およびその他の高レベルネットワークプロトコルより下で実行されます。

セキュリティードメイン (security domain)

PKI サブシステムの集中リポジトリーまたはインベントリー。その主な目的は、サブシステム間の信頼できる関係を自動的に確立することにより、新しい PKI サービスのインストールと設定を容易にすることです。

セルフテスト

インスタンスの起動時とオンデマンドの両方の Certificate System インスタンスをテストする機能。

対象暗号化 (symmetric encryption)

同じ暗号化キーを使用して特定のメッセージを暗号化および復号する暗号化方式。

署名アルゴリズム (signature algorithm)

デジタル署名の作成に使用される暗号化アルゴリズム。Certificate System は、MD5 および SHA 署名アルゴリズムをサポートしています。暗号アルゴリズム (cryptographic algorithm)、デジタル署名 (digital signature)も併せて参照してください。

署名付き監査ログ (signed audit log)

監査ログ (autdit log) を参照してください。

署名証明書

公開鍵がデジタル署名の作成に使用される秘密鍵に対応する証明書。たとえば、Certificate Manager には、発行する証明書の署名に使用する秘密鍵に対応する公開鍵を持つ署名証明書が必要です。

署名鍵

署名用途に使用する秘密鍵署名鍵とその同等の公開鍵、および 暗号鍵 (encryption key) とその同等の公開鍵は、デュアルキーペア (dual key pair) を構成します。

T

token key service (トークンキーサービス、TKS)

スマートカードの APDU およびトークン CUID などの他の共有情報に基づいて、スマートカードごとに特定の個別のキーを取得するトークン管理システムのサブシステム。

ツリー階層 (tree hierarchy)

LDAP ディレクトリーの階層構造。

トークン (token)

PKCS #11 モジュール の スロット (slot) に関連付けられたハードウェアまたはソフトウェアデバイス。暗号化サービスを提供し、必要に応じて証明書および鍵を保存します。

トークン処理システム (token processing system, TPS)

Enterprise Security Client とスマートカードを直接対話して、これらのスマートカードのキーと証明書を管理するサブシステム。

トークン管理システム (TMS)

相互に関連するサブシステム (CA、TKS、TPS、および任意で KRA) は、スマートカード (トークン) の証明書を管理するために使用されます。

信頼 (trust)

個人または他のエンティティーに確定します。公開鍵インフラストラクチャー (public-key infrastructure, PKI) では、信頼とは、証明書のユーザーと、その証明書を発行した 認証局 (certificate authority, CA) との間の関係を指します。CA が信頼されている場合は、その CA が発行する有効な証明書を信頼できます。

改ざんの検出 (tamper detection)

電子形式で受信したデータが同じデータの元のバージョンと完全に対応することを保証するメカニズム。

V

仮想プライベートネットワーク (virtual private network, VPN)
企業の地理的に離れた部署を接続する方法。VPN を使用すると、部署間は暗号化されたチャンネルを介して通信できるため、通常はプライベートネットワークに制限される認証済みの機密トランザクションが可能になります。

索引

シンボル

D

E

F

I

K

L

N

O

P

R

S

T

付録F 改訂履歴

改訂番号は本ガイドに関するものであり、Red Hat Certificate System のバージョン番号とは関係ありません。

改訂 10.1-1 **Mon Jan 25 2021** **Florian Delehaye**
さまざまな修正および改善点。

改訂 10.1-0 **Wed Dec 02 2020** **Florian Delehaye**
Red Hat Certificate System 10.1 ガイドのリリース

改訂 10.0-0 **Wed Sep 17 2020** **Florian Delehaye**
Red Hat Certificate System 10.0 ガイドのリリース